ΠΑΝΕΠΙΣΤΗΜΙΟ ΣΤΕΡΕΑΣ ΕΛΛΑΔΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗΝ ΒΙΟΙΑΤΡΙΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Χρήση του Περιβάλλοντος CLIPS για τη δημιουργία ενός Εμπειρου Συστήματος Λήψης Απόφασης στην Ιατρική

Καρυστιάνης Γιώργος
Αριθμός Μητρώου: 1

Επιβλέπουσα Καθηγήτρια
Δρ Παπαγεωργίου Ελπινίκη
Επ. Καθηγητρια (Π.Δ.407/80)

Λαμία , 2008

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την επιβλέπουσα της παρούσας διπλωματικής μου εργασίας την κα. Δρ Ελπινίκη Παπαγεωργίου για την ανάθεση του θέματος και την ευκαιρία να ασχοληθώ εις βάθος με αυτό. Έχοντας μια άριστη συνεργασία από την αρχή που συναντηθήκαμε, η πτυχιακή εργασία πραγματοποιήθηκε χωρίς πιέσεις φέρνοντας στην επιφάνεια ευχάριστα συναισθήματα κάνοντας την εκπόνηση αυτής της διπλωματικής περισσότερο χόμπι παρά μια υποχρεωτική δουλειά. Διπλό ευχαριστώ και για την πολύτιμη συμβουλή και υποστήριξη της στο ψυχολογικό υπόβαθρο.

Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου που πάντα με στήριζαν σε οποιαδήποτε προσπάθεια κατέβαλλα στην ζωή μου, την αδερφή μου που με βοήθα από την στιγμή που έμαθε ότι έχει έναν μικρό αδερφό καθώς και ένα ξεχωριστό άτομο που με στήριξε στην όλη μου αυτή προσπάθεια.

.

Λαμία, Σεπτέμβρης 2008

Καρυστιάνης Γιώργος

Περίληψη

Στην παρούσα διπλωματική εργασία με θέμα «Χρήση του περιβάλλοντος CLIPS για την δημιουργία ενός έμπειρου συστήματος λήψης απόφασης στην ιατρική» στόχος είναι η ανάπτυξη ενός έμπειρου συστήματος διάγνωσης αλλεργιών υλοποιημένο στη γλώσσα CLIPS (C Language Integrated Production System). Αρχικά γίνεται μία εκτενής παρουσίαση στα έμπειρα συστήματα και στα βασικά χαρακτηριστικά τους, στη συνέχεια αναλύεται διεξοδικά η δομή τους καθώς και τα πιθανά οφέλη που επιφέρει η χρήση τους στον τομέα της υγείας. Στην πορεία γίνεται εκτεταμένη αναφορά στη γλώσσα CLIPS, περιγράφοντας τα χαρακτηριστικά της, τις εντολές της και λειτουργίες της ώστε να μπορέσει ο ενδιαφερόμενος να κατανοήσει τον τρόπο δημιουργίας του συγκεκριμένου έμπειρου συστήματος. Έχοντας μιλήσει για την έννοια του έμπειρου συστήματος και τη σύνταξη της γλώσσας CLIPS, περιγράφεται το πρόβλημα για το οποίο έχει αναπτυχθεί το κατάλληλο σύστημα ενώ παρουσιάζεται και τεκμηριώνεται υπολογιστικά το πρόγραμμα που το υλοποιεί. Στο τέλος της παρούσας διπλωματικής εργασίας παρουσιάζονται: α) τα αποτελέσματα της λειτουργίας του έμπειρου συστήματος διάγνωσης αλλεργιών το οποίο ονομάζεται Διαγνωστικό Σύστημα Αλλεργιών ή χρησιμοποώντας την αγγλική ορολογία Diagnostic Allergy System (D.A.S), β) πραγματοποιείται μια συζήτηση σχετικά με τη συγκεκριμένη εφαρμογή και τον τρόπο απόδοσης του έμπειρου συστήματος για τη διάγνωση αλλεργιών και γ) καταγράφονται συγκεκριμένα συμπεράσματα που προέκυψαν σε σχέση με τη χρησιμότητα των έμπειρων συστημάτων στην ιατρική και την αξιολόγηση του εργαλείου σχεδιασμού έμπειρων συστημάτων CLIPS.

Λέξεις κλειδιά- Τεχνητή νοημοσύνη, Έμπειρο σύστημα, αναπαράσταση γνώσης, CLIPS, Αλλεργία, λήψη απόφασης

Abstract

The aim of this work with title “Use of the CLIPS for the development of an expert system for medical decision support” is to design and to develop an expert system implemented in the language CLIPS (C Language Integrated Production System) that might diagnose allergies from a set of symptoms. Firstly we present analytically the main aspects of expert systems as well as their basic characteristics with the analysis of their structure and their possible profits to the field of medicine which bring to light by their use. The most friendly and easy for use language in the design of expert systems is the CLIPS. In the second chapter we describe CLIPS’s characteristics, its commands and its functions in order for the reader to understand the way this particular expert system was created. With the detailed presentation of the expert systems and the language CLIPS, we continue by describing our problem for which the expert system is developed while we show and justified the program which implemented it. At the end of this work we evaluate the proposed expert system which was achieved with the import of certain medical cases and we summarize to the following aspects: a) presentation of results for the operation of this expert system called Diagnostic Allergy System (“D.A.S”) b) discussion for this particular application and for its perfomance based on the diagnosis of the allergies and c) assignment of certain logical observations which occurred as regard with the usefulness of the expert systems in medicine and the usefulness of the designing tool of expert systems called CLIPS.

Keywords— Artificial Intelligence, Expert System, knowledge representation, CLIPS, Allergy, decision making

Περιεχόμενα

Κεφαλαίο 1: Έμπειρα Συστήματα

1.1 Εισαγωγή

Από την αρχή σχεδόν της ύπαρξης των υπολογιστών, οι επιστήμονες ονειρεύτηκαν τη δημιουργία προηγμένων συστημάτων τα οποία θα μπορούσαν να αφομοιώσουν την ανθρώπινη γνώση και λογική. Από όλες τις μοντέρνες τεχνολογικές αναζητήσεις, η έρευνα για τη δημιουργία συστημάτων τεχνητής νοημοσύνης αποτελεί μια από τις πιο φιλόδοξες και πιο συναρπαστικές. Παρόλο που είχαν πραγματοποιηθεί προσπάθειες πριν από 30 χρόνια περίπου για την ανάπτυξη και την εφαρμογή ενός τέτοιου συστήματος στις ιατρικές επιστήμες, με τον καιρό το ενδιαφέρον μειωνόταν σταδιακά (K.S. Metaxiotis, J.-E. Samouilidis **2000**).

Η υποστήριξη λήψης απόφασης στην ιατρική είναι το τμήμα της κλινικής γνώσης και της πληροφορίας συσχετιζόμενης με τον ασθενή, φιλτραρισμένη έξυπνα ή παρουσιασμένη σε κατάλληλο χρόνο ώστε να ενισχύσει τη φροντίδα του ασθενή. Τα ιατρικά ιδρύματα συνεχώς υιοθετούν εργαλεία που θα προσφέρουν υποστήριξη στη λήψη απόφασης προκείμενου να βελτιώσουν τη διαχείριση των ασθενών και της προϋπάρχουσας γνώσης και να μειώσουν τα ιατρικά λάθη. Οι προμηθευτές και οι διαχειριστές φροντίδας μαζί με λίγη ή καθόλου εκπαίδευση στους υπολογιστές θα κληθούν να εκτιμήσουν, να επιλέξουν και να συμβάλλουν στην ανάπτυξη ενός συστήματος υποστήριξης λήψης απόφασης για την πρακτική τους.

Οι εφαρμογές της τεχνητής νοημοσύνης και άλλων υπολογιστικών και πληροφοριακών επιστημονικών τεχνικών στο πεδίο της ιατρικής έχουν φέρει ως αποτέλεσμα την ανάπτυξη κλινικών συστημάτων υποστήριξης λήψης αποφάσεων (Clinical Decision Support Systems) τα οποία καλούνται και έμπειρα συστήματα (http://www.openclinical.org/dssSuccessFactors.html).

Ένα έμπειρο σύστημα είναι ένα λογισμικό σύστημα που επιχειρεί να αναπαράγει την ερμηνεία ενός ή περισσότερων ανθρώπων, ειδικών σε συγκεκριμένους τομείς όντας μια εφαρμογή τεχνητής νοημοσύνης. Ένας πιο ανεπίσημος ορισμός για το τι είναι τα έμπειρα συστήματα δόθηκε από τους Wyatt [1] και Spiegelhalter [2] (K.S. Metaxiotis, J.-E. Samouilidis **2000**):

" Ένα έμπειρο σύστημα είναι ένα ενεργό σύστημα γνώσης που χρησιμοποιεί δυο ή περισσότερα αντικείμενα δεδομένων ασθενών για να παράγει συγκεκριμένη για την περίπτωση συμβουλή "

[1] Jeremy Wyatt: Καθηγητής του πανεπιστήμιου του Birmingham ο οποίος έχει έκδοση αρκετές δημοσιεύσεις που αφορούν έξυπνα συστήματα βασισμένα σε γνώση, μερικές μάλιστα σε συνεργασία με τον David John Spiegelhalter. Είναι διευθυντής του εργαστήριο ρομποτικής στο τμήμα της πληροφορικής του πανεπιστήμιου του Birmingham.

[2] David John Spiegelhalter: Στατιστικολόγος(γεννημένος στις 16 Αυγούστου του1953) που εργάζεται τώρα ως καθηγητής στο πανεπιστήμιο του Cambridge για την δημόσια κατανόηση του κινδύνου ενώ εργάζεται στη μονάδα βιοστατιστικής του Συμβουλίου ιατρικής έρευνας, στο ίδρυμα δημόσιας υγείας του πανεπιστήμιο του Cambridge. Ένας από τους κορυφαίους 250 αναγερμένους επιστήμονες στον κόσμο κατά τη διάρκεια των τελευταίων δέκα ετών.

Μια μεγάλη ποικιλία από μεθόδους μπορεί να χρησιμοποιηθεί για την προσομοίωση της γνώσης του ειδικού επιστήμονα με τις πιο γνωστές να είναι:

1. η δημιουργία της λεγόμενης Βάσης Γνώσης (Knowledge Base) που χρησιμοποιεί κάποιο φορμαλισμό αναπαράστασης γνώσης για να συλλέξει τη γνώση του ειδικού επιστήμονα (Subject Matter Experts knowledge-SME).

2. μια διαδικασία συλλογής εκείνης της γνώσης από το SME και κωδικοποιώντας την σύμφωνα με το φορμαλισμό που καλείται εφαρμοσμένη μηχανική γνώσης (Knowledge Engineering [3]).

Δυο απεικονίσεις πραγματικών έμπειρων συστημάτων μπορούν να μας δώσουν μια ιδέα για το πώς δουλεύουν (http://en.wikipedia.org/wiki/Expert_system). Στη μια περίπτωση, σε κάποιες χημικές εγκαταστάσεις καθαρισμού, ένας υπάλληλος ήταν έτοιμος να αποσυρθεί και η εταιρεία ανησυχούσε για την απώλεια του αφού δεν θα είχαν πια τις γνώσεις του στη διαχείριση ενός «πύργου» διαχώρισης συστατικών και αυτό θα συντελούσε σοβαρά σε εσωτερική ζημιά στην εταιρεία. Ανατέθηκε λοιπόν σε έναν μηχανικό γνώσης (knowledge engineer) να παράγει και να δημιουργήσει ένα έμπειρο σύστημα που θα περιέχει την ειδική γνώση του υπάλληλου και θα μπορεί να τη χρησιμοποιεί όταν τη χρειάζεται. Η άλλη περίπτωση είναι το σύστημα MYCIN που αναπτύχθηκε από τις γνώσεις ειδικών επιστημόνων με εμπειρία στις βακτηριακές μολύνσεις των οποίων η ερμηνεία θεωρήθηκε ότι ήταν ανώτερη από εκείνη του μέσου γιατρού. Η δεκαετία του '80 ήταν ο χρόνος της μέγιστης δημοτικότητας των έμπειρων συστημάτων, το ενδιαφέρον ωστόσο άρχισε να μειώνεται με ραγδαίο ρυθμό μετά από την αρχή του AI WINTER [4].

Οι Wyatt και Spiegelhalter ανέφεραν ότι τα τελικά στάδια ενός έμπειρου συστήματος θα πρέπει να περιλαμβάνουν εκτίμηση των αποτελεσμάτων πάνω στις διαδικασίες της υγείας και στα αποτελέσματα του ασθενή. Πάνω στην άποψη των Wyatt και Spiegelhalter έρχεται να προστεθεί και η πρόταση του Randolph A.Miller [5] ο οποίος πίστευε ότι(K.S. Metaxiotis, J.-E. Samouilidis **2000**):

"η τελική σκέψη της αξιολόγησης/ εκτίμησης ενός έμπειρου συστήματος είναι να διαπιστωθεί εάν ο χρηστής και το σύστημα μαζί είναι καλύτερα από τον αβοήθητο χρηστή σε σχέση με ένα συγκεκριμένο πρόβλημα"

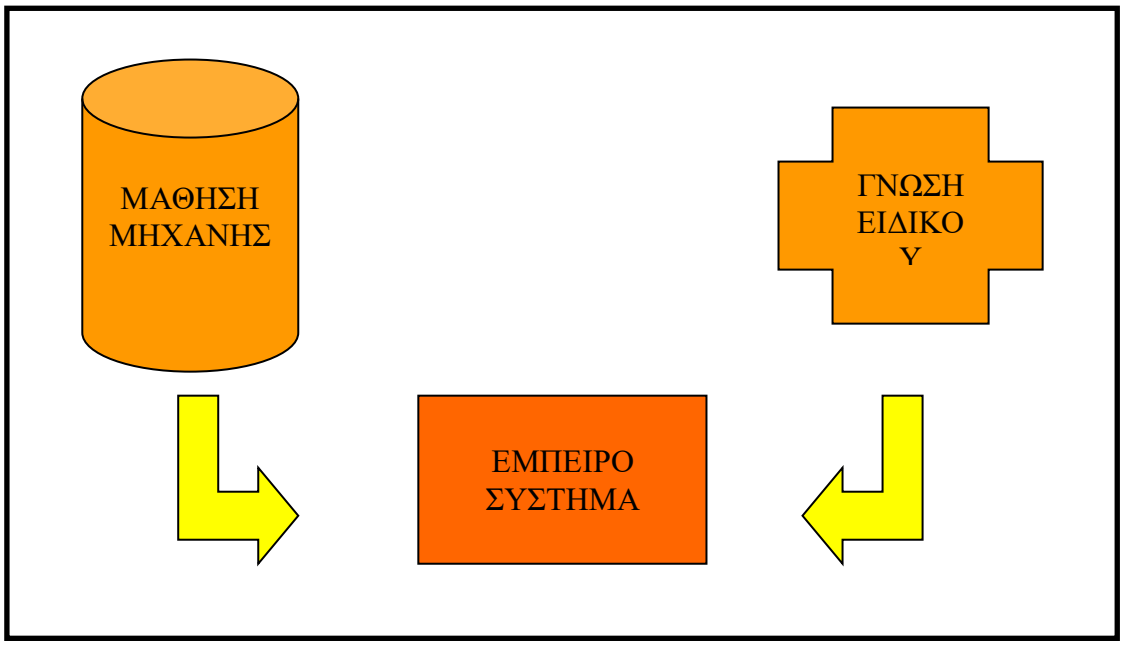
[3] Knowledge engineering (εφαρμοσμένη μηχανική γνώσης): όρος που αναφέρεται στο χτίσιμο, στην διατήρηση και την ανάπτυξη των βασισμένων σε γνώση συστημάτων. Έχει πολλά κοινά σημεία με την κατασκευή λογισμικού και σχετίζεται και με πολλά πεδία της επιστήμης των υπολογιστών όπως η τεχνητή νοημοσύνη, οι βάσεις δεδομένων, η εξόρυξη δεδομένων, τα έμπειρα συστήματα και τα συστήματα υποστήριξης λήψης απόφασης.

[4] AI WINTER: Είναι μια κατάρρευση που έγινε στον τομέα της ερευνάς στην τεχνητή νοημοσύνη. Ο όρος πλάσθηκε κατ' αναλογία προς την ανηλεή σπείρα ενός πυρηνικού χειμώνα: μια αλυσωτή αντίδραση της απαισιοδοξίας στην κοινότητα AI, που ακολουθήθηκε από την απαισιοδοξία στον Τύπο, που ακολουθήθηκε από μια αυστηρή μείωση στη χρηματοδότηση, που ακολουθήθηκε από το τέλος της έρευνας.

[5] Randolph A. Miller: Καθηγητής στο τμήμα βιοιατρικής πληροφορικής του πανεπιστήμιου Donald A.B. and Mary M. Lindberg. Είναι πολύ γνωστός για την εξειδίκευση του πάνω σε ιατρικά διαγνωστικά συστήματα και σε βάσεις γνώσεων. Εργάστηκε στο INTERNIST-I διαγνωστικό σχέδιο το 1973 ενώ το 1985, ξεκίνησε να δουλεύει στο Quick Medical Reference (QMR). Έχει βραβευτεί πολλές φορές για τις δουλείες του στα ιατρικά έμπειρα συστήματα και σήμερα θεωρείται ένα σημαντικό πρόσωπο πάνω στον τομέα των βιοιατρικής πληροφορικής.

1.2 Τι είναι τα έμπειρα συστήματα;

Πιο αναλυτικά, τα έμπειρα συστήματα στην ιατρική είναι προγράμματα υπολογιστών σχεδιασμένα για να βελτιώνουν την επεξεργασία και το αποτέλεσμα της κλινικής απόφασης. Χρησιμοποιώντας ευρέως τεχνικές τεχνητής νοημοσύνης, βοηθούν επαγγελματίες στον τομέα της υγείας για την επιλογή της σωστής διάγνωσης και θεραπείας όταν η πληροφορία-συνήθως τα δεδομένα του ασθενή, είναι αβέβαιη και ανολοκλήρωτη. Τα βασισμένα σε γνώση έμπειρα συστήματα χρησιμοποιούν ανθρώπινη γνώση προκείμενου να δώσουν λύση σε προβλήματα τα οποία υπό άλλες συνθήκες θα απαιτούσαν ανθρώπινη νοημοσύνη δηλαδή έμπειρους και ειδικούς επιστήμονες. Αναπαριστούν συγκεκριμένη γνώση σαν δεδομένα ή σαν κανόνες μέσα στο υπολογιστικό σύστημα και καλούνται όταν χρειαστεί για να δοθεί μια λύση σε ένα συγκεκριμένο πρόβλημα (Εικόνα 1.2.1).



Εικόνα 1.2.1 Από τι αποτελείται ένα έμπειρο σύστημα (Peter Lucas, Expert Systems: A knowledge-based approach to intelligent systems, Department of Information and Knowledge Systems, Institute of Computing and Information Sciences, University of Nijmegen, The Netherlands 2002)

Τα έμπειρα συστήματα διακρίνονται σε δυο κατηγορίες. Τα ενεργά και τα παθητικά. Τα παθητικά χρησιμοποιούνται όταν το κλινικό προσωπικό χρειάζεται βοήθεια. Ένα παράδειγμα είναι το Quick Medical Reference (QMR) [6], πρόγραμμα που λειτουργεί σαν ένα ενσωματωμένο εγχειρίδιο αναφοράς. Στα ενεργά συστήματα ο υπολογιστής δίνει συμβουλές όταν χρειάζεται και παραδείγματα τέτοιων εφαρμογών είναι οι προειδοποιήσεις και οι υπενθυμίσεις που στέλνονται. Παράδειγμα τέτοιων συστημάτων είναι το σύστημα HELP το οποίο είναι ενσωματωμένο σε ηλεκτρονικά ιατρικά αρχεία.

[6] Quick Medical Reference: Γνωστό και ως QMR αναπτύχθηκε με την βοήθεια του INTERNIST-I, και είναι μια πηγή πληροφόρησης σε βάθος που βοηθάει το κλινικό προσωπικό στην διάγνωση ενήλικων ασθενειών. Παρέχει ηλεκτρονική πρόσβαση σε περισσότερες από 750 ασθένειες αναπαριστώντας την πλειοψηφία των διαταραχών που διαπιστώνονται σε καθημερινή εξάσκηση καθώς την επίτομη των λιγότερων γνωστών ασθενειών.

Ένα αποτελεσματικό έμπειρο σύστημα πρέπει να έχει ακριβή δεδομένα, φιλική επιφάνεια διεπαφής με το χρήστη, αξιόπιστη βάση δεδομένων και καλό μηχανισμό εξαγωγής συμπερασμάτων. Η επιφάνεια εργασίας χρήστη είναι το συστατικό του συστήματος που γεφυρώνει την επικοινωνία του συστήματος με το χρήστη. Η βάση είναι αυτή που περιέχει τη γνώση του συστήματος όπως το είδος των ρίσκων, τους κανόνες για τις ασθένειες και τα αποτελέσματα. Η μηχανή παρεμβολής παρομοιάζεται με τις ικανότητες του υπολογιστή να λύνει τα προβλήματα-ξέρει πως και ποτέ να εφαρμόσει την κατάλληλη γνώση. Αυτοί οι μηχανισμοί εξαγωγής συμπερασμάτων μπορούν να κωδικοποιηθούν με διάφορες τεχνικές για να αναπαραστήσουν την γνώση όπως είναι οι πιθανότητες και τα σημασιολογικά δίκτυα [7].

Τα έμπειρα συστήματα όπως προαναφέρθηκε παραπάνω είναι υποκατηγορία εφαρμογών τεχνητής νοημοσύνης που αποτελεί κλάδο ακαδημαϊκής έρευνας για πάνω από τριάντα χρόνια. Γενικεύοντας τα έμπειρα συστήματα χαρακτηρίζονται κυρίως από:

1. Συμβολική λογική παρά από αριθμητικούς υπολογισμούς.
2. Οδηγούμενες από δεδομένα (data driven) διαδικασίες.
3. Την ικανότητα για εξήγηση των τελικών συμπερασμάτων με ιδέες οι οποίες είναι σημαντικές για τον χρήστη.
4. Μια πολύ συγκεκριμένη βάση δεδομένων που είναι κατανοητή από έναν ειδικό του πεδίου στο οποίο βασίζεται αυτή η βάση.

Σήμερα που η εποχή έχει πλήρως αποδεχτεί την τεχνολογία και τα επιτεύγματα της, τα έμπειρα συστήματα μπορούν να εφαρμοστούν σε αρκετούς τομείς όπως στον έλεγχο της κυκλοφορίας, στην υποστήριξη αποφάσεων, στη διαχείριση, στην αυτόματη πλοήγηση σκαφών, στη διάγνωση μηχανολογικών βλαβών, στη ιατρική διάγνωση και στην βιοπληροφορική.

1.2.1 Δομή έμπειρων συστημάτων

Πιο αναλυτικά, ένα έμπειρο σύστημα αποτελείται από τέσσερα στοιχεία (K.S. Metaxiotis, J.-E. Samouilidis **2000**):

- Από τη **βάση γνώσης** (knowledge base) που είναι η καρδιά κάθε έμπειρου συστήματος και περιέχει μια αναπαράσταση ανθρώπινης γνώσης και εμπειρίας (σε μορφή κανόνων, λογικών εκφράσεων).

[7] Σημασιολογικό δίκτυο (semantic network): Είναι ένα κατευθυνόμενο ή μη-κατευθυνόμενο γράφημα που αποτελείται από κόμβους οι οποίοι αναπαριστούν ιδέες και από ακμές που αντιπροσωπεύουν τις σημασιολογικές σχέσεις μεταξύ των ιδεών. Χρησιμοποιείται συχνά ως ένας τρόπος αναπαράστασης γνώσης.

- Από τη **μηχανή συμπεράσματος ή μηχανή εξαγωγής συμπερασμάτων** (inference engine) που χρησιμοποιείται κατά την διάρκεια της χρονικής περιόδου που ζητείται η συμβουλή. Εξετάζει τη κατάσταση της βάσης, ελέγχει το περιεχόμενο της και καθορίζει τη σειρά με την οποία θα γίνουν οι ενέργειες για την εξαγωγή του συμπεράσματος-αποτελέσματος.

- Από τη **μνήμη εργασίας** (working memory) η οποία περιέχει όλα τα αποτελέσματα της διαδικασίας εξαγωγής συμπερασμάτων κατά τη διάρκεια της χρονικής περιόδου που ζητείται η συμβουλή.

- Από τη **διεπαφή εισόδου εξόδου** (input output interface) που δίνει τη δυνατότητα στο χρήστη να προμηθεύσει γεγονότα και δεδομένα στο σύστημα και να πραγματοποιήσει ερωτήσεις ή να δώσει συμβουλή και επεξηγήσεις.

Η γνώση που υπάρχει ενσωματωμένη στη βάση μπορεί να είναι είτε **αφηρημένη** (abstract) είτε **συγκεκριμένη** (concrete). Η αφηρημένη γνώση αποτελείται από ένα σύνολο αντικειμένων και ένα σύνολο κανόνων που καθορίζει τις συσχετίσεις μεταξύ αντικειμένων. Η συγκεκριμένη γνώση παρέχει πληροφορία αναφορικά με κάποια συγκεκριμένη εφαρμογή. Για παράδειγμα στην ιατρική διάγνωση, τα συμπτώματα, οι ασθένειες και οι συσχετίσεις μεταξύ τους αποτελούν την αφηρημένη γνώση, ενώ τα συμπτώματα κάποιου ασθενή συνιστούν τη συγκεκριμένη γνώση. Η συγκεκριμένη γνώση αναφέρεται στα γεγονότα (facts) που υπάρχουν για μια συγκεκριμένη εφαρμογή. Ο τύπος της γνώσης αυτής είναι δυναμικός, δηλαδή μπορεί να μεταβληθεί από εφαρμογή σε εφαρμογή και για τον λόγο αυτό αποθηκεύεται στη μνήμη εργασίας. Η αφηρημένη γνώση είναι σταθερή αφού δεν αλλάζει από εφαρμογή σε εφαρμογή και αποθηκεύεται στη βάση γνώσης ενώ η συγκεκριμένη γνώση είναι εφήμερη, δηλαδή δεν αποτελεί συστατικό του μόνιμου τμήματος του συστήματος και καταστρέφεται μετά τη χρησιμοποίησή της. Σε αιτιοκρατικές καταστάσεις, οι σχέσεις μεταξύ ενός συνόλου αντικειμένων αναπαριστώνται με ένα σύνολο κανόνων.

Τα έμπειρα συστήματα ποικίλλουν ανάλογα με τον βαθμό πολυπλοκότητας τους, τη λειτουργία τους και τις εφαρμογές τους. Επίσης διαφέρουν σε σχέση με τις πρακτικές κατευθύνσεις (practice guidelines) και με τα κλινικά μονοπάτια στα οποία ζητούν να γίνει εισαγωγή των δεδομένων του ασθενή με συγκεκριμένες μεταβλητές με αποτέλεσμα να παρέχουν συγκεκριμένες προτάσεις για την κάθε περίπτωση (Robert Trowbridge & Scott Weingarten **2000**). Ανάμεσα στις πιο κοινές μορφές των έμπειρων συστημάτων στην ιατρική είναι αυτά που καθορίζουν τη δόση των φάρμακων, όντας βασισμένα σε υπολογιστή μετά τη εισαγωγή των δεδομένων του ασθενή. Τέτοια συστήματα είναι ιδιαίτερα χρήσιμα στον έλεγχο της διαχείρισης των φαρμάκων με ένα στενό θεραπευτικό εγχειρίδιο Α-Ω. Πιο πολύπλοκα συστήματα περιλαμβάνουν υπολογιστικά διαγνωστικά εργαλεία όπου παρόλο που διακρίνεται μια συγκεκριμένη εισαγωγή δεδομένων για τον κάθε ασθενή, μπορεί να χρησιμεύσουν σαν μία βάση για εκείνον τον ασθενή που δείχνει να παρουσιάζει μια συγχυσμένη συλλογή συμπτωμάτων μη ξεκάθαρης διάγνωσης. Άλλα συστήματα είτε περίπλοκα είτε απλά, παρέχουν

υπενθυμίσεις σχετικά με την κατάλληλη διαχείριση βασισμένα σε προηγούμενα δεδομένα. Αυτά μπορεί να είναι και τα πιο πρακτικά εάν ενωθούν με ηλεκτρονικά αρχεία ασθενών και με συγκεκριμένη υπολογιστική σειρά εισαγωγής. Όμως εάν χρησιμοποιηθούν καθημερινά, τα έμπειρα συστήματα μπορούν να παρέχουν στους γιατρούς προτάσεις για την πιο κατάλληλη φροντίδα, μειώνοντας έτσι την πιθανότητα ιατρικών λαθών και βελτιώνοντας το επίπεδο των παρεχόμενων υπηρεσιών υγείας. Τα έμπειρα συστήματα συλλέγουν μικρά κομματάκια γνώσης και τα ενσωματώνουν σε μια βάση συλλογής δεδομένων που χρησιμοποιείται ώστε να λύσει ένα πρόβλημα μέσω της κατάλληλης γνώσης (που υπάρχει μέσα σε αυτήν). Ένα διαφορετικό πρόβλημα μπορεί να αντιμετωπιστεί χρησιμοποιώντας το ίδιο πρόγραμμα χωρίς επαναπρογραμματισμό.

Για την κατασκευή των έμπειρων συστημάτων χρησιμοποιούνται συγκεκριμένα εργαλεία. Τα έμπειρα συστήματα μπορούν να κατασκευαστούν με τη χρήση ενός ή δύο τύπων εργαλείων, τις **γλώσσες τεχνητής νοημοσύνης** και τα λεγόμενα **κελύφη** (shells) έμπειρων συστημάτων (K.S. Metaxiotis, J.-E. Samouilidis **2000**). Με διαφορά οι πιο κοινές γλώσσες τεχνητής νοημοσύνης που χρησιμοποιούνται στην ανάπτυξη ιατρικών έμπειρων συστημάτων είναι η LISP (List Processing Language) [8] και η PROLOG [9]. Χρησιμοποιώντας αυτές τις γλώσσες, ο σχεδιαστής του συστήματος έχει έναν αρκετά υψηλό βαθμό ελευθέριας όσον αφορά την επιλογή του για τις τεχνικές αναπαράστασης γνώσης και για τις στρατηγικές έλεγχου. Βέβαια η χρήση τέτοιων γλωσσών απαιτεί έναν υψηλό βαθμό εξειδίκευσης και ικανότητας.

Τα κελύφη είναι ειδικά λογισμικά εργαλεία που προσπαθούν να συνδυάσουν την ευελιξία των γλωσσών τεχνητής νοημοσύνης με την αποτελεσματικότητα του κόστους και να παρέχουν πιο γενικές καταστάσεις ανάπτυξης. Πιο αναλυτικά ένα κέλυφος είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τη σχεδίαση και τη διατήρηση εφαρμογών βασισμένων σε γνώση. Παρέχει μια βήμα προς βήμα μεθοδολογία για τον μηχανικό γνώσης (knowledge engineer) [10] που επιτρέπει σε ειδικούς από διάφορους τομείς να περιλαμβάνονται άμεσα στη δομή και στη κωδικοποίηση της γνώσης (Εικόνα 1.2.1.1). Τα περισσότερα έμπειρα συστήματα αναπτύσσονται με τα κελύφη. Είναι εξοπλισμένα με έναν μηχανισμό παρεμβολής και απαιτούν συγκεκριμένη γνώση που θα εισέλθει σύμφωνα με ένα συγκεκριμένο τύπο (format).

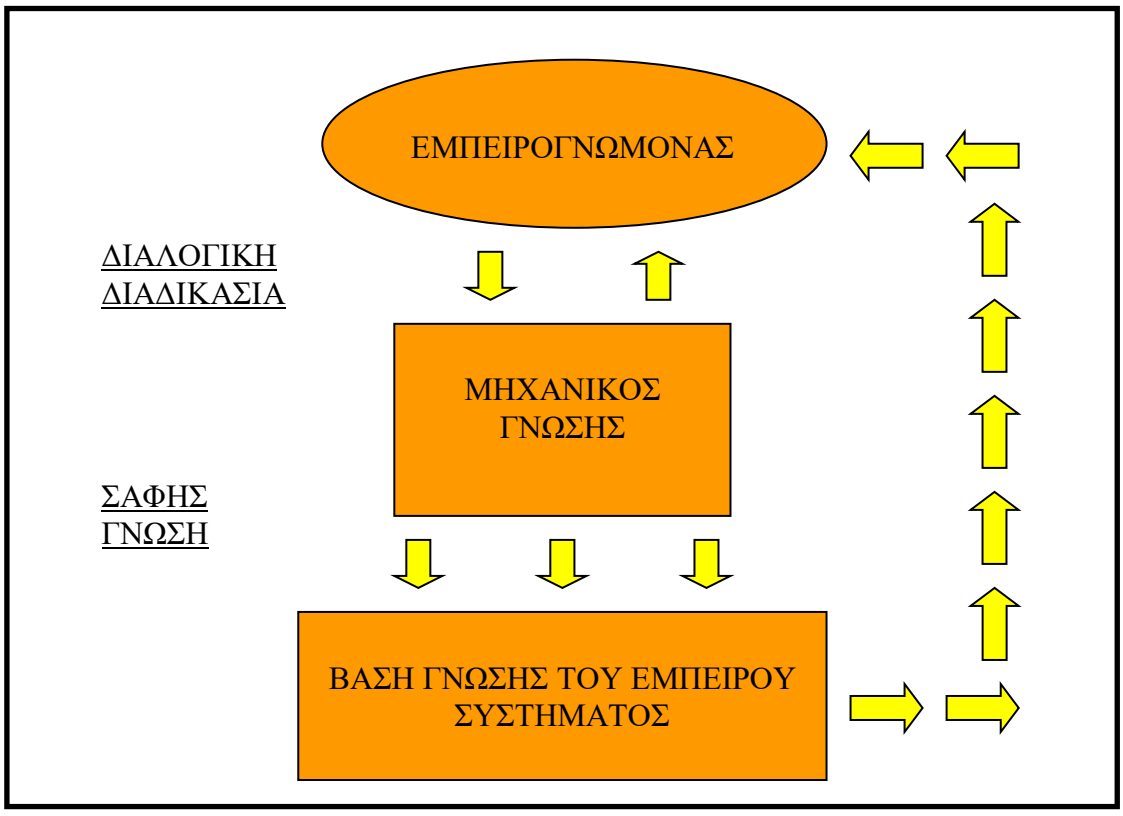
8] LISP: Οικογένεια από γλώσσες προγραμματισμού με αρκετά πολύπλοκη σύνταξη. Είναι η δεύτερη πιο παλιά γλώσσα υψηλού προγραμματισμού με την εμφάνιση της να πραγματοποιείται γύρω στο 1958. Σήμερα έχει ευρεία χρήση και αποδοχή και μόνο η Fortran είναι πιο παλιά(γιαγιά πλέον!). Το όνομα της προέρχεται από το όρο "List Processing Language". Οι συνδεδεμένες λίστες είναι μια από τις δομές δεδομένων που υλοποιεί ως γλώσσα ενώ ο πηγαίος κώδικας της φτιάχνεται από λίστες. Η Lisp έχει ελαφρώς τροποποιηθεί από τότε που εμφανίστηκε για πρώτη φορά και ένας αριθμός από διαλέκτους έχουν προκύψει. Στη σημερινή εποχή οι πιο κοινοί διάλεκτοι είναι η Common Lisp και η Scheme. Δημιουργήθηκε σαν μαθηματική σημείωση για προγράμματα υπολογιστών και ύστερα χρησιμοποιήθηκε για ερευνά πάνω στη τεχνητή νοημοσύνη.

[9] Prolog: Γλώσσα λογικού προγραμματισμού. Είναι μια γλώσσα γενικού σκοπού συσχετιζόμενη συχνά με την τεχνητή νοημοσύνη. Δημιουργήθηκε στις αρχές της δεκαετής του ’70 στην Γαλλία. Είναι μια από τις πρώτες γλώσσες λογικού προγραμματισμού και παραμένει μέχρι και σήμερα στο πάνθεον των πιο δημοφιλών γλωσσών προγραμματισμού. Στόχος ήταν η επεξεργασία της φυσικής γλώσσας αλλά στην πορεία ο σκοπός της επεκτάθηκε σε αλλά πεδία όπως τα έμπειρα συστήματα και τα βιντεοπαιχνίδια.

[10] Μηχανικός γνώσης (Knowledge engineer): Είναι ειδικοί υπολογιστικών συστημάτων εκπαιδευμένοι στο πεδίο των έμπειρων συστημάτων. Σκοπός τους η επιλογή του λογισμικού του έμπειρου συστήματος, η πρόσληψη γνώσης του ειδικού επιστήμονα και η αναπαράσταση της στη βάση γνώσης του έμπειρου συστήματος.

Έχουν πολλά χαρακτηριστικά όπως εργαλεία για έγγραφη σε υπερκείμενο, για κατασκευή φιλικών επιφανειών εργασίας χρήστη, για χειραγώγηση λιστών, συμβολοσειρών και αντικειμένων και για επικοινωνία με εξωτερικά προγράμματα και βάσεις δεδομένων.

[



Εικόνα 1.2.1.1 Ανάπτυξη ενός έμπειρου συστήματος (Χρυσόστομος Στύλιος, Έμπειρα συστήματα, Τεχνολογικό Εκπαιδευτικό Ίδρυμα Ηπείρου, 2005)

1.3 Σκοπός των έμπειρων συστημάτων

Τα έμπειρα συστήματα μέχρι στιγμής είναι σχεδιασμένα έτσι ώστε να προσφέρουν τις πολύτιμες υπηρεσίες τους για τη υποστήριξη αποφάσεων στην διάγνωση, στη διαχείριση φαρμάκων και ασθενειών και στην πραγματοποίηση προληπτικών επεμβάσεων. Οι δραστηριότητες τους χωρίζονται σε τέσσερις βασικές κατηγορίες:

- Διαχειρίζονται τη κλινική πολυπλοκότητα και τις ιατρικές λεπτομέρειες κρατώντας τους ασθενείς σε ερευνητικά και χημειοθεραπευτικά πρωτόκολλα ενώ ανιχνεύουν εντολές και παραπομπές που επακολουθούν καθώς και πρωτόκολλα προληπτικής φροντίδας.

- Ελέγχουν το κόστος παρακολουθώντας τις ιατρικές συνταγές και αποφεύγοντας τις πολύπλοκες και αχρείαστες εξετάσεις.

- Υποστηρίζουν τη κλινική διάγνωση και τις επεξεργασίες θεραπευτικών πλάνων. Προωθούν τη χρήση των καλύτερων εξασκήσεων, των κλινικών μονοπατιών σε συγκεκριμένες συνθήκες και τη βασισμένη στον ανθρώπινο πληθυσμό διαχείριση.

- Ενισχύουν τη υποστήριξη κλινικής κωδικοποίησης και καταγραφής μέσω της έγκρισης των διαδικασιών και των όποιων παρεμβάσεων.

Πιο αναλυτικά όμως τα έμπειρα συστήματα εκτελούν ένα ευρύ φάσμα λειτουργιών οι οποίες συνοψίζονται παρακάτω:

1. **Παραγωγή συναγερμών και υπενθυμίσεων**. Ειδοποιούν, υπενθυμίζουν και ενημερώνουν τον χρήστη για τυχόν αλλαγές στη κατάσταση του ασθενή. Ένα παράδειγμα είναι η σύνδεση τους σε ένα μόνιτορ μέσω του οποίου παρακολουθείται η υγεία του ασθενή. Επιπλέον, πραγματοποιούν σάρωση των αποτελεσμάτων των εργαστηριακών τεστ και υπενθυμίζουν εάν κάτι σημαντικό δεν έχει πραγματοποιηθεί ή έχει ξεχαστεί να γίνει.

2. **Διαγνωστική βοήθεια**. Παρέχουν βοήθεια στην διάγνωση ενός ασθενή όταν η κατάσταση του είναι περίπλοκη ή σπάνια ή το άτομο που κάνει τη διάγνωση είναι άπειρο, με πιθανές διαγνώσεις βασιζόμενες πάνω στα δεδομένα του ασθενή.

3. **Καθορισμός και σχεδιασμός θεραπείας**. Προσδιορίζουν μια θεραπεία με βάση τις ιδιαιτερότητες του ασθενή ή διορθώνουν προς το καλύτερο μια ήδη υπάρχουσα θεραπεία.

4. **Μόρφωση και εκπαίδευση**. Χρησιμοποιούνται για να εκπαιδεύουν και για να δίνουν την δυνατότητα σε γιατρούς να εξασκηθούν σε διάφορους ιατρικούς στόχους (tasks).

1.4 Πλεονεκτήματα

Από τη χρήση των ιατρικών έμπειρων συστημάτων προκύπτουν συγκεκριμένα οφέλη τα οποία διακρίνονται σε τρεις κατηγορίες (E. Coiera **2003**):

- Η βελτίωση ασφάλειας του ασθενή είναι ένα από τα πιο σημαντικά πλεονεκτήματα που προσφέρονται με τη εφαρμογή των έμπειρων συστημάτων στη ιατρική. Με τη σωστή και ορθολογική χρήση τους υπάρχει μείωση των ιατρικών λαθών, βελτίωση της δοσοληψίας αλλά και της επιλογής φαρμάκων καθώς και των εξετάσεων.

- Παροχή καλύτερης ποιότητας φροντίδα με διάθεση κλινικού χρόνου από το ιατρικό προσωπικό για άμεση παροχή φροντίδας υγείας με στόχο τη βέλτιστη ικανοποίηση του ασθενή.

- Διαπιστώνεται βελτίωση αποτελεσματικότητας στη παροχή υγείας με μείωση του κόστους διαμέσου γρηγορότερης επεξεργασίας διαγνώσεων, με μείωση αντίστροφων γεγονότων και με φθηνότερα φάρμακα ωστόσο εξίσου αποτελεσματικά.

Όμως θα μπορούσε κανείς να προσθέσει τα πλεονεκτήματα της χρήσης των έμπειρων συστημάτων αφορούν τόσο το ιατρικό προσωπικό που είναι οι τελικοί χρήστες όσο και τον ίδιο τον ασθενή που είναι ο τελικός αποδέκτης. Εκτός λοιπόν από τις τρεις μεγάλες κατηγορίες που αναφέρθηκαν παραπάνω, μια πιο αναλυτική προσέγγιση αξίζει να ειπωθεί. Με την βοήθεια της τεχνολογίας που εφαρμόζεται στον τομέα της ιατρικής με την μορφή των έμπειρων συστημάτων διακρίνεται ότι (http://www.pcai.com/web/ai_info/expert_systems.html,http://www.openclinical.org/dssSuccessFactors.html):

- Υπάρχει μεγάλο όφελος από άποψη εξόδων σε μεγάλες εταιρείες. Με τη χρησιμοποίηση των μικρών συστημάτων σώζονται χιλιάδες δολάρια ενώ μέσω των μεγάλων υπάρχει καθαρό κέρδος από άποψη κόστους μέχρι και σε εκατοντάδες εκατομμύρια δολάρια. Είναι θέμα χρόνου η βελτίωση ποιότητας η οποία πλέον θα θεωρηθεί κίνητρο για την άμεση λειτουργία έμπειρων συστημάτων σε μεγάλες εταιρείες, επιχειρήσεις και μεγάλα νοσοκομειακά κέντρα.
- Περιορίζεται η πιθανότητα επιλογής της λάθος απόφασης και ενισχύεται η βελτίωση της ποιότητας λήψης αποφάσεων.
- Διατηρείται η γνώση των ειδικών οι οποίοι αποχωρούν από το χώρο λόγω σύνταξης ή άλλων προσωπικών και μη, λόγων. Κατά αυτόν τον τρόπο όχι μόνο διατηρείται αλλά συλλέγεται η εξατομικευμένη γνώση από διάφορους τομείς και συγκεντρώνεται σε ένα σημείο χωρίς να παρατηρείται απώλεια.
- Παρέχεται οποιαδήποτε κλινική πληροφορία όποια στιγμή χρειάζεται, όταν χρειάζεται, με ευκολία πρόσβασης του ιατρικού προσωπικού.
- Πραγματοποιείται εισαγωγή σε νέα προϊόντα και υπηρεσίες όπως είναι τα εργαλεία υλοποίησης (λογισμικό κλπ) των έμπειρων συστημάτων και τα αποτελέσματα τους.
- Γίνεται επιτάχυνση της ανθρώπινης κατανόησης, αντίδρασης και αλληλεπίδρασης.
- Υπάρχει η δυνατότητα να εφαρμοστεί κλινικός έλεγχος, το λεγόμενο audit ή clinical audit [11], μέσω του οποίου αναλύονται τα αποτελέσματα από την χρήση τους με σκοπό να αποφευχθούν στο μέλλον τυχόν λάθη και να επιτευχθεί το μέγιστο και καλύτερο επίπεδο υπηρεσιών.

[11] Clinical audit: Είναι ο λεγόμενος κλινικός έλεγχος. Ο επίσημος ορισμός δόθηκε το 1993 μέσω της Εθνικής Υπηρεσίας Υγείας (National Health Service) της Αγγλίας και καθορίζεται ως μια διαδικασία βελτίωσης ποιότητας που αναζητά να βελτιώσει την φροντίδα του ασθενή καθώς και τα αποτελέσματα μέσω συστηματικού σχολιασμού της φροντίδας ενάντια καθορισμένων κριτηρίων και με την χρήση/ εφαρμογή της αλλαγής. Το συστατικό-κλειδί ενός κλινικού audit είναι ότι το αποτέλεσμα ελέγχεται ξανά και ξανά έτσι ώστε αυτό που πρέπει να γίνει, γίνεται και αν δεν γίνεται, παρέχει μια αξιόπιστη βάση για να γίνουν στο μέλλον πιθανές βελτιώσεις.

- Υποστηρίζεται η έρευνα για καλύτερα και πιο ενθαρρυντικά αποτελέσματα και συστήματα κάτι στο οποίο συμβάλλει και ο κλινικός έλεγχος.
- Υποστηρίζεται η ιατρική εκπαίδευση και μόρφωση του κλινικού προσωπικού χωρίς μεγάλο κόστος και με ευκολία μάθησης δίχως να υπάρχει δυσκολία προσαρμογής.

1.5 Πιθανά μειονεκτήματα

Εκτός από τα πολλά και σημαντικά οφέλη που κάνουν τα έμπειρα συστήματα ένα εργαλείο βελτίωσης των συνθηκών στην κλινική πράξη, θα ήταν λογικό κατά την εφαρμογή τους να προκύπτουν κάποια (πιθανά) μειονεκτήματα (http://www.openclinical.org/dssSuccessFactors.html):

- Θεωρούνται ότι δεν είναι ευέλικτα αφού χρησιμοποιούν συγκεκριμένους κανόνες και λογική και δεν έχουν την κρίση ενός έμπειρου γιατρού. Κατά συνέπεια εάν δεν διαθέτουν και το τελευταίο κομμάτι της εξειδικευμένης γνώσης, είναι πολύ πιθανό για πολύ ιδιαίτερες περιπτώσεις ασθενών, να γίνει λάθος διάγνωση.
- Η υπερβολική πίστη στη τεχνολογία ίσως οδηγήσει σε σκοτεινά «μονοπάτια» στην ιατρική. Ο ρόλος της τεχνολογίας στην ιατρική είναι να βοηθάει το κλινικό προσωπικό και όχι να το αντικαθιστά.
- Εντείνεται η σκέψη του κλινικού προσωπικού και αντικαθίσταται με την λογική ενός υπολογιστή.
- Περιορίζεται η εμπειρία και η φαντασία του κλινικού προσωπικού, με τον υπολογιστή να αντικαθιστά τον άνθρωπο σε θέματα που απαιτούν κρίση και συναίσθημα.
- Τα ηθικά και τα νομικά ζητήματα που προκύπτουν είναι ασταθή ακόμη, πόσο μάλλον στη περίπτωση που γίνει κάποιο ιατρικό λάθος
- Υπάρχει πιθανότητα να μην προσαρμοστεί άμεσα το ιατρικό προσωπικό τόσο από πλευράς εκπαίδευσης και μάθησης όσο και από πλευράς χειρισμού, διαδικασία η οποία μπορεί να αποβεί χρονοβόρα και εξαιρετική δύσκολη.
- Υπάρχει δυσκολία εκτίμησης τους αφού τα αποτελέσματα δεν είναι εγγυημένα. Αυτό είναι ένα σημαντικό μειονέκτημα εφόσον τα ιατρικά έμπειρα συστήματα χαρακτηρίζονται από αβεβαιότητα και ασάφεια τόσο στους κανόνες που ακολουθούν και στα πλαίσια τα οποία εφαρμόζουν που υπάρχουν στη βάση δεδομένων όσο και από τη εγκυρότητα της απάντησης του χρήστη.
- Έχουν υψηλό κόστος συντήρησης, ενημέρωσης και εκπαίδευσης.

Με βάση τα πλεονεκτήματα και τα μειονεκτήματα των έμπειρων συστημάτων θα μπορούσε κανείς να πει ότι η αποδοχή τους στον τομέα της υγείας εξαρτάται από πολλούς σημαντικούς και ιδιαίτερους παράγοντες (http://www.openclinical.org/dssSuccessFactors.html). Ο πρώτος είναι το κόστος. Η έρευνα για το σχεδιασμό, τη ανάλυση, τη δημιουργία και τη εφαρμογή ενός

έμπειρου συστήματος απαιτεί πολυάριθμους (κυρίως) χρηματικούς πόρους σε σημείο απαγορευτικό. Εδώ να σημειωθεί ότι έστω και εάν πραγματοποιηθεί σωστή έρευνα με κατάλληλη διαχείριση πόρων, το κόστος εκπαίδευσης και μάθησης του ιατρικού προσωπικού θεμελιώνεται ως ένα βασικό εμπόδιο για τη περαιτέρω χρήση τους. Τα νομικά και τα ηθικά ζητήματα που προκύπτουν κατά τη χρήση τους στον τομέα υγείας είναι ένας άλλος παράγοντας. Μέχρι να βρεθούν απαντήσεις σε φλέγοντα ερωτήματα της ιατρικής κοινότητας αλλά και για τους ασθενείς δεν μπορούν να χρησιμοποιηθούν ευρέως. Για παράδειγμα εάν προκύψει ένα σφάλμα στη διάγνωση του ασθενή και χορηγηθεί λάθος φάρμακο μέσω του έμπειρου συστήματος ποιος θα είναι ο υπεύθυνος; Ο σχεδιαστής του έμπειρου συστήματος που έκανε το λάθος ενώ υποθετικά δεν θα έπρεπε να το έχει κάνει; Ο γιατρός που άκουσε το σύστημα είτε είναι έμπειρος είτε όχι; Μήπως σε αυτό συντελούν και οι νοσηλευτές; Παρατηρείται λοιπόν πως τέτοια φαινόμενα θα διακρίνονται συχνά και μέχρι να αναπτυχθούν ολοκληρωτικά και να ξεκαθαριστούν οι ρόλοι ανάμεσα στο ιατρικό προσωπικό, δεν γίνεται να γίνει πλήρη εφαρμογή και αποδοχή των έμπειρων συστημάτων και δικαιολογημένα. Επιπλέον, ο βαθμός αποδοχής τους από το ιατρικό προσωπικό διαδραματίζει ένα σημαντικό παράγοντα καθώς θα πρέπει πρώτα να εξεταστεί η διεπαφή ανάμεσα στον χρηστή (κλινικό προσωπικό) και στον υπολογιστή. Εάν αυτή είναι εύχρηστη, φιλική και χωρίς πολυπλοκότητες τότε εδραιώνεται μια καλή έως αρμονική σχέση συνεργασίας χωρίς προβλήματα. Ωστόσο εάν θα υπάρξει διαθεσιμότητα συντήρησης και ενημέρωσης των συστημάτων θα συμβάλλει στη μάλλον άμεση ενσωμάτωση τους στο κλινικό (και όχι μόνο) περιβάλλον. Τέλος ο χρόνος και ο τύπος της εκπαίδευσης θα συμβάλλει στη χρησιμοποίηση τους από τη ιατρική κοινότητα και από τον κύκλο των ασθενών. Με όλα τα παραπάνω που αναφέρθηκαν θα διαπιστωθεί το ποσοστό της απόδοσης τους στο μέλλον και κατά πόσο μπορούν να βοηθήσουν στην ιατρική.

Εκτός βέβαια από τους παράγοντες οι οποίοι θα καθορίσουν την αποδοχή των έμπειρων συστημάτων, ήδη υπάρχοντα εμπόδια ορθώνονται για τη κυριαρχία τους στην αγορά της υγείας. Πρέπει να σημειωθεί ότι κάθε ασθενής είναι συγκεκριμένη-αν όχι ιδιαίτερη, περίπτωση και υπάρχει η πιθανότητα εάν γίνουν διαθέσιμα τα συγκεκριμένα ιατρικά δεδομένα, να γίνει λάθος με αποτέλεσμα την δημιουργία ηθικών ζητημάτων και αβέβαιων ρίσκων ενώ δεν υπάρχει και η κατάλληλη χρηματοδότηση για να γίνει περαιτέρω ερευνά (K.S. Metaxiotis, J.-E. Samouilidis **2000**). Τα παραπάνω μπορεί να προκύψουν κατά την διάρκεια λειτουργίας ενός έμπειρου συστήματος αλλά όμως δεν είναι μη αντιμετωπίσιμα. Γι' αυτό ακριβώς τον λόγο χρησιμοποιούνται ασαφής λογική [12] και BELIEF δίκτυα [13] (K.S. Metaxiotis, J.-E. Samouilidis **2000**).

[12] Ασαφής λογική: Αφορά την αναπαράσταση ανακριβούς ή αδιευκρίνιστης γνώσης, στηρίζεται στη θεωρία των ασαφών συνόλων και χρησιμοποιεί την έννοια του βαθμού συμμετοχής/ αλήθειας και όχι τον κάθετο διαχωρισμό αλήθεια- ψευδός. Ασχολείται με την ασάφεια της γνώσης και όχι με την τυχαιοτητά της.

[13] BELIEF δίκτυο ή δίκτυο Bayesian: Πιθανοκρατικό γραφικό μοντέλο που αναπαριστά ένα σετ μεταβλητών. Για παράδειγμα, ένα δίκτυο Bayesian θα μπορούσε να αναπαραστήσει τις πιθανοκρατικές σχέσεις ανάμεσα στις ασθένειες και στα συμπτώματα. Με δοσμένα συμπτώματα, το δίκτυο μπορεί να χρησιμοποιηθεί με στόχο τον υπολογισμό των πιθανοτήτων της παρουσίας διάφορων ασθενειών.

1.6 Γιατί είναι σημαντικά τα έμπειρα συστήματα;

Στη σημερινή εποχή όπου διακρίνονται κορυφαίες τεχνολογικές εξελίξεις στον τομέα της υγείας, έχουν ήδη αναπτυχθεί και εφαρμοστεί έμπειρα συστήματα σε πολλά πεδία της ιατρικής όπως στη διάγνωση και θεραπεία ασθενειών. Κάποιοι όμως ισχυρίζονται ότι τα έμπειρα συστήματα χρησιμοποιούνται και εφαρμόζονται λάθος, ωστόσο κανείς δεν μπορεί να αμφισβητήσει το γεγονός ότι είναι σημαντικά. Αντιθέτως υποστηρίζεται ότι είναι χρήσιμα για δυο βασικούς λόγους (K.S. Metaxiotis, J.-E. Samouilidis **2000**):

- Χρησιμεύουν στην υποστήριξη απόφασης ώστε να υπενθυμίζουν σε έναν έμπειρο γιατρό τις επιλογές που υπάρχουν και τα θέματα που πρέπει να σκεφτεί. Τέτοια συστήματα εφαρμόζονται κυρίως στην ιατρική.

- Παίζουν σημαντικό ρόλο στη λήψη αποφάσεων. Κατά αυτόν τον τρόπο παρέχεται βοήθεια σε άτομα χωρίς μεγάλη εμπειρία ώστε να έχουν τη δυνατότητα να πάρουν αποφάσεις που είναι πολύ μπροστά από το πεδίο τους ή το εύρος γνώσεων τους.

Ένα καλό στοιχείο των έμπειρων συστημάτων είναι ότι υπερτερούν σε σχέση με τα παραδοσιακά πληροφοριακά συστήματα αφού κατέχουν (K.S. Metaxiotis, J.-E. Samouilidis **2000**):

- **Διαθεσιμότητα:** Οι ειδικοί δεν γεννιούνται αλλά δημιουργούνται μέσα από εκπαίδευση και εξάσκηση. Είναι γνωστό ότι για να αποκτήσει κάποιος εξειδίκευση σε ένα συγκεκριμένο τομέα πρέπει τουλάχιστον να ασχολείται 5 χρόνια. Σε αντίθεση με τον άνθρωπο τα έμπειρα συστήματα έχουν όλη την εξειδίκευση ενσωματωμένη μέσα τους ενώ ποτέ δεν θα κουραστούν ή θα αποσυρθούν παρά μόνο εάν το θελήσει ο χρήστης.

- **Συνέπεια:** Ακόμα και οι καλύτεροι στον τομέα τους ειδικοί μπορεί να κάνουν λάθη ή να ξεχάσουν κάτι σημαντικό.

- **Συνεκτικότητα:** Ενώ ένας ειδικός έχει ένα συγκεκριμένο φάσμα γνώσεων, τα έμπειρα συστήματα μπορούν να αφομοιώσουν γνώσεις και δεδομένα από αλλά πεδία προσφέροντας έτσι τη δυνατότητα περισσότερων επιλογών και δυνατοτήτων.

1.7 Παραδείγματα και εφαρμογές

Τη πρώτη δεκαετία της εφαρμογής της τεχνητής νοημοσύνης στην ιατρική, τα περισσότερα έμπειρα συστήματα αναπτύχθηκαν για να βοηθήσουν το κλινικό προσωπικό στη διαδικασία της διάγνωσης. Πολλά από αυτά ωστόσο δεν κατάφεραν να ξεφύγουν από το στάδιο του σχεδιασμού και της ερευνάς. Παρόλα αυτά, κάποια ξεχώρισαν και μεταμορφώθηκαν σε μέρος των εκπαιδευτικών συστημάτων. Μερικά σημαντικά παραδείγματα αναφέρονται παρακάτω:

- **MYCIN**: Είναι μέχρι στιγμής το πιο γνωστό από όλα (και όχι μόνο) που έχει αναπτυχθεί σε ικανοποιητικό βαθμό. Δημιουργήθηκε από το Πανεπιστήμιο του Stanford στα μέσα της δεκαετίας 1970 από τον Ted Shortliffe ως μια ερευνητική προσπάθεια για να παρέχει βοήθεια σε γιατρούς στην διαδικασία της διάγνωσης και της θεραπείας κάποιων συγκεκριμένων ασθενειών όπως η μηνιγγίτιδα. Αργότερα η χρησιμοποίηση του επεκτάθηκε και σε άλλες μολυσματικές ασθένειες.

- **PUFF:** Αναπτύχθηκε το 1979, χρησιμοποιώντας το EMYCIN κέλυφος. Σκοπός του η ασχολία με μέτρα που σχετίζονται με αναπνευστικά τεστ τα οποία αναγνωρίζουν πνευμονικές διαταραχές.

- **INTERNIST:** Είναι ένα από τα πρώτα συστήματα υποστήριξης λήψης απόφασης και αναπτύχθηκε το 1974 από το Πανεπιστήμιο του Pittsburgh. Σκοπός του είναι η διάγνωση της πλειοψηφίας των ασθενειών που συσχετίζονται με το πεδίο της εσωτερικής (internal) ιατρικής. Υποτίθεται ότι θεωρεί όλους τους πιθανούς ανασυνδυασμούς των ασθενειών που μπορούν να παρουσιαστούν στον ασθενή. Από την αρχή της δεκαετίας του 1980, αναγνωρίστηκε ότι το πιο πολύτιμο κομμάτι του συστήματος αυτού ήταν η ιατρική βάση δεδομένων που διέθετε ενσωματωμένη μέσα του. Επίσης χρησιμοποιήθηκε σαν βάση για μετέπειτα συστήματα όπως το Quick Medical Reference (QMR).

- **LLIAD:** Αναπτύσσεται εδώ και κάποια χρόνια (από τη δεκαετία του 1990), κυρίως για διάγνωση στην εσωτερική ιατρική. Σήμερα καλύπτει πάνω από 1500 διαγνώσεις στον τομέα αυτό βασιζόμενο σε χιλιάδες ευρήματα. Η τωρινή χρήση του είναι ως εργαλείο εκπαίδευσης σε φοιτητές της ιατρικής.

- **Dxplain**: Αναπτύχθηκε στο γενικό νοσοκομείο της Μασαχουσέτης το 1987. Συμμετέχει στη διαδικασία της διάγνωσης παίρνοντας ένα σύνολο από συμπτώματα και εργαστηριακά δεδομένα. Έπειτα παράγει μια λίστα με τις πιθανές διαγνώσεις. Χρησιμοποιείται καθημερινά σε πολλά νοσοκομεία και σε ιατρικές σχολές κυρίως για κλινική εκπαίδευση.

- **HELP**: Είναι ένα ολοκληρωμένο και βασιζόμενο στη γνώση νοσοκομειακό σύστημα πληροφόρησης που ξεκίνησε να λειτουργεί το 1980 και αναπτύχθηκε από το τμήμα πληροφορικής με εφαρμογές στην ιατρική (Medical Informatics) του Πανεπιστήμιου της Utah. Παρέχει στους γιατρούς προειδοποιήσεις και υπενθυμίσεις, ερμηνεύοντας δεδομένα ασθενών εφόσον υπάρχουν εγκαταστάσεις διάγνωσης και με προτάσεις για διαχείριση ασθενών. Λειτουργεί μέχρι στιγμής σε έξι μεγάλα νοσοκομεία στην Utah και σε αρκετές περιοχές των Η.Π.Α.

1.8 Ο ρόλος των έμπειρων συστημάτων στην κλινική φροντίδα

Στη σημερινή εποχή, η ερευνά για τα έμπειρα ιατρικά συστήματα πραγματοποιείται κυρίως από ιατρικούς ερευνητές και ακαδημαϊκούς σε ένα πολύ μικρό ποσοστό. Επειδή τα έμπειρα συστήματα στην ιατρική είναι μια αληθινή δύναμη και όχι ένα απλώς τυπικό και αδιάφορο θέμα, η τεχνολογία των έμπειρων συστημάτων στον τομέα της υγείας θα πρέπει με κάποιο τρόπο να εδραιωθεί. Επίσης θα πρέπει να γίνει η αναζήτηση κοινών κλινικών προβλημάτων τα οποία θα αντισταθμίζουν τον μεγάλο όγκο των κλινικών καθηκόντων (K.S. Metaxiotis, J.-E. Samouilidis **2000**).

Ωστόσο το να εστιάσει κανείς στην υποστήριξη της διαδικασίας διαχείρισης ασθενή βοηθάει να μην στρέφεται η προσοχή προς υπό-θέματα διάγνωσης. Η υποστήριξη διαχείρισης ασθενή οφείλει να πραγματοποιηθεί σε πρωτοβάθμια και δευτεροβάθμια ιδρύματα και όχι απλώς στα μεγάλα παραδοσιακά νοσοκομεία. Επιπλέον, η ανάπτυξη εφαρμογών για ένα ευρύ φάσμα κλινικών χρηστών από τις νοσηλευτές και τους γιατρούς σε διαιτολόγους, φαρμακοποιούς και άλλους επαγγελματίες στο χώρο της υγείας είναι επίσης μια καλή αλλαγή για την εδραίωση των έμπειρων συστημάτων.

Ο ρόλος τους στη φροντίδα υγείας είναι ξεκάθαρος. Βασισμένος στις προτάσεις που έγιναν παραπάνω, οι τομείς όπου μπορούν να εφαρμοστούν τα έμπειρα συστήματα είναι οι εξής (K.S. Metaxiotis, J.-E. Samouilidis **2000**):

- ✓ **Συστήματα εργαστηρίου**: Έχει αποδειχτεί από τα κλινικά εργαστήρια ότι είναι ένα άμεσο πεδίο εφαρμογής και χρήσης των έμπειρων συστημάτων.

- ✓ **Συστήματα συμβουλής φάρμακων**: Εδώ προτείνεται ο σχεδιασμός έμπειρων συστημάτων τα οποία θα βοηθήσουν το κλινικό προσωπικό με τη συνταγογράφηση φάρμακων και την επιλογή των πιο αποτελεσματικών από άποψη κόστους θεραπειών για τον κάθε ασθενή.

- ✓ **Ερμηνεία σήματος**: Η ανάπτυξη των συναγερμών και προειδοποιήσεων για αληθινού χρόνου κλινικά σήματα σε ιατρικές

μονάδες όπως η μονάδα εντατικής φροντίδας θα προσφέρει επιπλέον βοήθεια ταυτόχρονα με την υπηρεσία της κλινικής επαγρύπνησης.

- ✓ **Διασφάλιση ποιότητας**: Θα υπάρξει η ανάγκη να γίνει έλεγχος σε όλους τους τύπους των βασιζόμενων σε γνώση συστημάτων για να ενημερωθούν και να συμφωνούν με τα τρέχον πρότυπα.

- ✓ **Μόρφωση και εκπαίδευση**: Η ανάγκη για συνεχή εκπαίδευση κλινικών επαγγελματιών και ασθενών προσφέρει σημαντικές ευκαιρίες για αυτοματοποιημένη βοήθεια, χωρίς ιδιαίτερη δυσκολία. Ήδη, η εκπαίδευση για τους περισσότερους γιατρούς είναι μια διαδικασία που βρίσκεται σε εξέλιξη. Είναι σημαντικό να σημειωθεί ότι τα έμπειρα συστήματα θα έπρεπε να σχεδιάζονται και να υλοποιούνται με τον παραπάνω τύπο υποστήριξης.

1.9 Στόχος της εργασίας

Στόχος της διπλωματικής εργασίας με τίτλο «Χρήση του περιβάλλοντος CLIPS για τη δημιουργία ενός έμπειρου συστήματος λήψης απόφασης στην ιατρική» είναι η σχεδίαση και η ανάπτυξη ενός έμπειρου συστήματος διάγνωσης αλλεργιών υλοποιημένο στη γλώσσα CLIPS. Το ενδιαφέρον κομμάτι είναι ο τρόπος με τον οποίο τα έμπειρα συστήματα μπορούν να αναπαραστήσουν την ανθρώπινη γνώση και να την διαχειριστούν προκειμένου να βοηθήσουν σε προβλήματα λήψης απόφασης και να συμβάλλουν γενικότερα στη βελτίωση της παροχής υπηρεσιών φροντίδας υγείας. Σκοπός δεν είναι η δημιουργία ενός τέλειου συστήματος διάγνωσης αλλεργιών αλλά η υλοποίηση και η λειτουργία ενός τέτοιου συστήματος που να δείχνει παράλληλα ότι τα έμπειρα συστήματα είναι απλά, κατανοητά, χρήσιμα και ευέλικτα για τον υποψήφιο χρήστη. Πόλο έλξης αποτέλεσε και η δημιουργία του έμπειρου συστήματος το οποίο μέσω ερωτήσεων προς το χρηστή κάνει διάγνωση και προτείνει αντίστοιχη θεραπεία ενώ προσοχή τράβηξε και η μάθηση μιας καινούργιας γλώσσας προγραμματισμού.

Έχοντας στο μυαλό τη δημιουργία ενός έμπειρου συστήματος διάγνωσης αλλεργιών, αυτό που απασχόλησε περισσότερο είναι η πιστότητα που πρέπει να έχει το σύστημα από άποψη συμπτωμάτων αλλεργιών και προτεινόμενων θεραπειών. Προκειμένου να κυμανθεί το σύστημα σε πιο ρεαλιστικά επίπεδα, υπάρχει η ικανότητα διάγνωσης ιλαράς ή γρίπης επειδή κάποια συμπτώματα των προηγούμενων μοιάζουν με κάποια των αλλεργιών. Κύριος στόχος εδώ είναι να αποφευχθεί η σύγχυση και να μπορέσει ο χρηστής να λάβει την σωστή διάγνωση καθώς και την σωστή θεραπεία για τον ασθενή ώστε να μην υπάρξουν δυσάρεστες συνέπειες.

Το έμπειρο σύστημα διάγνωσης αλλεργιών καλείται «D.A.S» και η υλοποίηση του πραγματοποιήθηκε με τη βοήθεια ενός βασικού εργαλείου σχεδιασμού έμπειρων συστημάτων που λέγεται CLIPS (C Language Integrated Production System). Ο λόγος που επιλέχθηκε αυτό το εργαλείο είναι η ευκολία εκμάθησης του καθώς η σύνταξη κανόνων πραγματοποιείται

χωρίς πολυπλοκότητα ενώ θεωρείται και το πιο δημοφιλές εργαλείο σχεδίασης και υλοποίησης έμπειρων συστημάτων.

Από τη σχεδίαση, υλοποίηση και λειτουργία του έμπειρου συστήματος διάγνωσης αλλεργιών «D.A.S» θα διαπιστωθεί ότι τα έμπειρα συστήματα μπορούν να συνεισφέρουν θετικά σε εκπληκτικό ποσοστό στον τομέα της ιατρικής βελτιώνοντας της υπηρεσίες υγείας και μειώνοντας τη πιθανότητα να προκύψουν ιατρικά λάθη. Αυτό θα παρατηρηθεί από τη λειτουργία του «D.A.S» το οποίο θα μπορεί να προσφέρει υποστήριξη στη λήψη απόφασης στο κλινικό προσωπικό ενώ παράλληλα θα συμμετέχει και στη διαδικασία υποβοήθησης λήψης απόφασης στη περίπτωση που το προσωπικό είναι άπειρο ή αβέβαιο για τις επιλογές του φανερώνοντας και έναν εκπαιδευτικό χαρακτήρα.

Κεφάλαιο 2: Περιγραφή της γλώσσας CLIPS

2.1 Εισαγωγή

Η CLIPS είναι ένα εργαλείο σχεδιασμένο για τη δημιουργία προγραμμάτων και εφαρμογών στην περιοχή των έμπειρων συστημάτων. Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, ένα έμπειρο σύστημα είναι ένα περιβάλλον σχεδιασμένο με τέτοιο τρόπο ώστε να μοντελοποιεί την ανθρώπινη γνώση. Το όνομα CLIPS είναι ένα ακρωνύμιο του C Language Integrated Production System. Η σύνταξη και το όνομα αυτής της γλώσσας είναι εμπνευσμένα από το OPS (Official production system) του Charles Forgy [14] [15]. Οι πρώτες εκδοχές της CLIPS αναπτύχθηκαν περίπου το 1985 στο διαστημικό κέντρο Johnson της NASA μέχρι τα μέσα της δεκαετίας του 1990 όπου οι έρευνες για την τεχνολογία έμπειρων συστημάτων σταμάτησαν. Η πρώτη επίσημη έξοδος της γλώσσας ήταν το 1986 (http://en.wikipedia.org/wiki/Clips).

Η CLIPS καλείται εργαλείο έμπειρων συστημάτων επειδή είναι ένα ολοκληρωμένο περιβάλλον για την ανάπτυξη τους και περιλαμβάνει χαρακτηριστικά όπως ο ενσωματωμένος συντάκτης κειμένου editor και το εργαλείο αποσφαλμάτωσης. Η λέξη κέλυφος (shell) είναι το κομμάτι της CLIPS που πραγματοποιεί συμπερασμούς. Το κέλυφος της Clips παρέχει τα βασικά στοιχεία ενός έμπειρου συστήματος όπως (Joseph C. Giarratano **2002**):

- ✓ Την λίστα γεγονότων (fact list)
- ✓ Την λίστα στιγμιότυπων (instance list)
- ✓ Την βάση γνώσης (knowledge base) που περιέχει όλους τους κανόνες οι οποίοι αποτελούν την βάση κανόνων (rule base)
- ✓ Την μηχανή εξαγωγής συμπερασμάτων (inference engine)

[14] Το OPS5 (Official Production System) είναι μια βασισμένη σε κανόνες γλώσσα όντας η πρώτη που χρησιμοποιήθηκε σε ένα πετυχημένο έμπειρο σύστημα, το R1/XCON που εξακρίβωνε VAX υπολογιστές. Αναπτύχθηκε στα τέλη της δεκαετίας του 1970 από τον Charles Forgy. Χρησιμοποιεί πρόσθια αλυσίδωση παρεμβολής μηχανής. Τα προγράμματα εκτελούνται σκανάροντας "στοιχεία μνήμης εργασίας" και ψάχνοντας για ταιριάσματα με τους κανόνες στην μνήμη.

[15] Γεννημένος στις 12 Δεκεμβρίου του 1949 στο Τέξας, ο καθηγητής πληροφορικής Charles L.Forgy είναι γνωστός για την ανάπτυξη του αλγόριθμου Rete που χρησιμοποιήθηκε στην επίσης δικιά του προγραμματιστική γλώσσα OPS5 και για άλλες γλώσσες συστημάτων παραγωγής που είχαν ως στόχο την δημιουργία έμπειρων συστημάτων

Η CLIPS είναι ένα από τα πιο διαδεδομένα εργαλεία για την δημιουργία έμπειρων συστημάτων επειδή είναι γρήγορη, εύκολη στη χρήση, ευέλικτη και παρέχεται δωρεάν. Αναβαθμίζεται και υποστηρίζεται συχνά από τον αυθεντικό συγγραφέα Gary Riley[16] ακόμα και τώρα που παρέχεται δωρεάν. Ενσωματώνει μια αντικειμενοστραφή προσέγγιση όσον αφορά την γλώσσα προγραμματισμού ονομαζόμενη COOL[17] που χρησιμοποιείται για την κατασκευή έμπειρων συστημάτων. Ενώ είναι γραμμένη σε C, η διεπαφή χρήστη μοιάζει περισσότερο με εκείνη της προγραμματιστικής γλώσσας LISP. Προεκτάσεις μπορούν να γραφτούν σε C και η CLIPS μπορεί να καλεστεί και μέσω της C (http://en.wikipedia.org/wiki/Clips). Άλλα εργαλεία για την ανάπτυξη έμπειρων συστημάτων είναι το Jess (Java Expert System Shell-όντας μια μηχανή εξαγωγής συμπερασμάτων γραμμένη σε Java), η γλώσσα Prolog και η γλώσσα Forth.

2.2 Σκοπός της CLIPS

Η CLIPS σχεδιάστηκε με σκοπό να αναπτύξει ένα λογισμικό με το οποίο θα μπορεί να αναπαρασταθεί και να διαχειριστεί η ανθρώπινη γνώση. Ένα πρόγραμμα γραμμένο σε τέτοια γλώσσα μπορεί να αποτελείται από **κανόνες** (rules), από **γεγονότα** (facts) και από **αντικείμενα** (objects). Ένα βασισμένο σε κανόνες έμπειρο σύστημα γραμμένο σε CLIPS μπορεί να είναι ένα οδηγούμενο από δεδομένα πρόγραμμα όπου τα γεγονότα και τα αντικείμενα εάν επιθυμείται είναι τα δεδομένα που διεγείρουν την εκτέλεση με την βοήθεια της μηχανής εξαγωγής συμπερασμάτων .

Για το πώς η CLIPS διαφέρει από άλλες γλώσσες όπως η Pascal, η Ada, η Basic, η Fortran και η C ας δώσουμε ένα παράδειγμα. Διαδικαστική γλώσσα είναι μια γλώσσα προγραμματισμού που απαιτεί προγραμματιστική πειθαρχία. Δηλαδή οι προγραμματιστές που χρησιμοποιούν τέτοιες γλώσσες πρέπει να αναπτύξουν μια κατάλληλη σειρά από εντολές έχοντας ως στόχο την επίλυση του προβλήματος βασιζόμενοι στην επεξεργασία και στον προγραμματισμό των δεδομένων. Σε αυτές τις γλώσσες, η εκτέλεση μπορεί να πραγματοποιηθεί χωρίς δεδομένα. Αυτό γιατί οι δηλώσεις είναι επαρκείς σε τέτοιες γλώσσες ώστε να μπορούν να προκαλέσουν εκτέλεση (http://www.techweb.com/encyclopedia/defineterm.jhtml?term=PROCEDURAL+LANGUAGE). Πιο αναλυτικά, μια δήλωση όπως print 2+2 θα πρέπει να εκτελείται άμεσα από την Basic. Αυτή είναι μια ολοκληρωμένη δήλωση που δεν απαιτεί καμία επιπλέον προσθήκη δεδομένων για να γίνει η εκτέλεση. Παρόλα αυτά στην CLIPS τα δεδομένα απαιτούνται να προκαλέσουν την εκτέλεση των κανόνων. Αρχικά η CLIPS είχε ικανότητες για την αναπαράσταση μόνο κανόνων και γεγονότων.

[16] Ο Gary Riley εργαζόταν στη NASA για πάνω από 10 χρόνια. Στα χρόνια εργασίας του εκεί ήταν υπεύθυνος για τον σχεδιασμό και την ανάπτυξη ενός μέρος της Clips που θα βασιζόταν σε κανόνες. Από την στιγμή που έφυγε από την NASA συνεχίζει ανεξάρτητα να αναπτύσσει και να διατηρεί την Clips.

[17] Η Cool είναι μια μικρή γλώσσα σχεδιασμένη για χρήση σε μια προπτυχιακή εργασία ενός μαθήματος compiler. Παρόλο που είναι μικρή για μια εργασία, διαθέτει πολλά χαρακτηριστικά των σημερινών μοντέρνων γλωσσών προγραμματισμού όπως αντικείμενα και αυτόματη διαχείριση μνήμης.

Εντούτοις, οι βελτιώσεις της έκδοσης 6.0 επιτρέπουν σε κανόνες να ταιριάζουν αντικείμενα καθώς και γεγονότα (Joseph C. Giarratano **2002**). Επίσης, αυτά μπορούν να χρησιμοποιηθούν χωρίς κανόνες μέσω αποστολής μηνυμάτων και έτσι η μηχανή εξαγωγής συμπερασμάτων δεν είναι απαραίτητη εάν χρησιμοποιούμε μόνο αντικείμενα. Μιλώντας πιο συγκεκριμένα η CLIPS περιέχει **γεγονότα** (facts), **κανόνες** (rules), **συναρτήσεις** (deffunctions) και **γενικές συναρτήσεις** (generic functions) καθώς και **αντικειμενοστραφή προγραμματισμό**. Στην CLIPS, ο βαθμός προτεραιότητας (salience) επιτρέπει σε ένα χρήστη να δώσει βάρος ή προτεραιότητα σε ένα κανόνα. Καλό θα ήταν να πούμε πως η γλώσσα LISP έχει επηρεάσει την ανάπτυξη της CLIPS.

2.3 Περιγραφή της γλώσσας CLIPS

2.3.1 Γενική περιγραφή

Υπάρχει ένα έτοιμο περιβάλλον της CLIPS και σε windows αλλά και σε linux για τους χρήστες της. Συγκεκριμένα στο περιβάλλον windows υπάρχει το εκτελέσιμο αρχείο clips.exe που παρέχει το περιβάλλον μαζί με το συντάκτη και τη μηχανή εξαγωγής συμπερασμάτων προκειμένου να μπορεί να χρησιμοποιηθεί από κάποιον άμεσα. Η CLIPS ξεκινάει με την κατάλληλη εντολή στο σύστημα. Θα πρέπει να εμφανίζεται :

CLIPS>

Δίπλα στην εντολή CLIPS> ο χρηστής έχει την ικανότητα να εισάγει τις εντολές, να καλεί κανόνες, να θέτει νέα γεγονότα. Επίσης έχει την ικανότητα να γράφει εντολές απευθείας στη CLIPS. Αυτή η “mode” στην οποία εισάγονται εντολές κατά αυτόν τον τρόπο λέγεται **top-level** (ανώτερου επιπέδου). Για να πραγματοποιηθεί έξοδος από τη CLIPS αρκεί να πληκτρολογηθεί η εντολή **(exit)**.

2.3.2 Γεγονότα

Αναφέρθηκε παραπάνω ότι η CLIPS χρειάζεται γεγονότα και κανόνες για να λειτουργήσει. Πιο συγκεκριμένα ένα γεγονός είναι μια λίστα από σύμβολα που περικλείεται από παρενθέσεις.

Παράδειγμα

(name DR DOOM)

Κάθε γεγονός έχει μοναδικό αριθμό-ταυτότητα **(fact index)** που καθορίζεται αυτόματα από την CLIPS όταν εισάγεται για πρώτη φορά (Ι. Βλαχάβας και συνεργάτες **2006**).

2.3.2.1 Εισαγωγή γεγονότων

Όπως και με άλλες προγραμματιστικές γλώσσες, η CLIPS αναγνωρίζει κάποιες συγκεκριμένες λέξεις κλειδιά. Έτσι για την εισαγωγή γεγονότων στη λίστα γεγονότων χρησιμοποιείται η εντολή (**assert**):

Παράδειγμα

- ✓ Εισαγωγή τριών γεγονότων:
  - **(assert (thor))**
  - **(assert (dr doom))**
  - **(assert (I have silver hair!))**

  ή
  - **(assert (thor)(dr doom)(I have silver hair!))**

Αυτό που θα φανεί στο παράθυρο εντολών (command window) είναι <fact-0> <fact-1> <fact-2> με τα νούμερα 0,1 και 2 να αντιστοιχούν στα γεγονότα “thor”, “dr doom” και “I have silver hair!”. Δηλαδή θα εμφανιστεί στην οθόνη:

<fact-0>thor
<fact-1>dr doom
<fact-2>I have silver hair!
for a total of 3 facts.
CLIPS>

Η CLIPS αυτόματα θα ονομάσει τα γεγονότα με αριθμούς χρησιμοποιώντας έναν σειριακά αυξανόμενο αριθμό που ονομάζεται “**fact index**” όταν ένα ή περισσότερα γεγονότα θα εισαχθούν (Joseph C. Giarratano **2002**). Εάν θελήσει κανείς να πραγματοποιήσει εισαγωγή γεγονότων με κενά τότε αυτά θα πρέπει να βρίσκονται μέσα σε εισαγωγικά αλλιώς η CLIPS δεν θα τα αναγνωρίσει.

Παράδειγμα

(assert(handsome-is “george”)
(assert(handsome-is “ george”)
(assert(handsome-is “george ”)
(assert(handsome-is “ george ”)

και έτσι διακρίνονται τέσσερα διαφορετικά γεγονότα. Ομοίως, εάν κάποιος θελήσει να εισάγει εισαγωγικά μέσα σε ένα γεγονός αρκεί να γράψει:

(assert(‘’\“george\”’’).

Αξιοσημείωτο είναι το γεγονός ότι στη CLIPS δεν γίνεται να εισαχθούν τα ίδια γεγονότα πάνω από μια φορά και εάν γίνει μία τέτοια προσπάθεια η CLIPS βγάζει το μήνυμα λάθους (false) (ωστόσο υπάρχει εντολή που το επιτρέπει **(self-fact-duplication)**) (Joseph C. Giarratano **2002**). Εάν όμως θελήσει κάποιος να εισάγει γεγονότα που είναι σχετικά μεταξύ τους κάνοντας ομαδοποίηση όπως για παράδειγμα μια λίστα ανθρώπων που άφησαν εποχή, τότε χρησιμοποιούμε την εντολή **(deffacts)**:

Παράδειγμα

(deffacts important-people "they change the history in their fields"
(Michael Jackson)
(Michael Jordan)
(J.F Kennedy)
(Martin Luther King))

Εδώ πραγματοποιείται εισαγωγή στην βάση γεγονότων, των γεγονότων "Michael Jackson", "Michael Jordan", "J.F Kennedy" και "Martin Luther King". Όταν τα γεγονότα εισέρχονται στη βάση γνώσης αυτόματα τους ανατίθεται ο αριθμός εισαγωγής (**fact-index**). Με το **(deffacts)** διακρίνεται ένας τρόπος αρχικοποίησης της βάσης γεγονότων (group of facts). Επίσης με τη χρησιμοποίηση αυτής της εντολής θα εισαχθεί και το "initial fact" μαζί με τα υπόλοιπα γεγονότα ενώ υπάρχει επιλογή προσθήκης κάποιου σχολίου δίπλα από το όνομα της ομάδας πάντα μέσα σε εισαγωγικά.

Όταν εισάγεται η εντολή **(reset)** τότε εμφανίζεται:

f-0 (initial-fact)
f-1 (Michael Jackson)
f-2 (Michael Jordan)
f-3 (J.F Kennedy)
f-4 (Martin Luther King)

- ✓ Ειδικό γεγονός αποτελεί το αρχικό γεγονός **(initial fact)**

Αυτό εισέρχεται όταν ενεργοποιηθεί η εντολή (reset) και έχει αριθμό ταυτότητας πάντα 0 (εξού και ο ορισμός αρχικό γεγονός)

2.3.2.2 Αφαίρεση γεγονότων

Αφαίρεση γεγονότων γίνεται με την εντολή **(retract)** με δομή εντολής (retract αριθμό- γεγονότος- θα- απομακρυνθεί). Από τη βάση γνώσης το γεγονός με αυτόν τον αριθμό ταυτότητας, θα απομακρυνθεί. Να σημειωθεί ότι εάν αφαιρείται ένα γεγονός, οι αριθμοί του "index" άλλων γεγονότων δεν αλλάζουν-όπως γίνεται στο ποδόσφαιρο. Όταν ένας παίχτης παίρνει κόκκινη κάρτα οι υπόλοιποι συνεχίζουν να παίζουν και δεν αλλάζουν νούμερα στο σύστημα. Επίσης μπορούν να αφαιρεθούν πολλά γεγονότα ταυτόχρονα αρκεί να γραφτούν οι αριθμοί των γεγονότων διαδοχικά ο ένας δίπλα στον άλλο ή και όλα εάν πληκτρολογηθεί **(retract *).**

Με την εντολή **(clear)** απομακρύνονται όλα τα γεγονότα από τη μνήμη και από τη λίστα γεγονότων. Πιο αναλυτικά, η CLIPS επανέρχεται στην αρχική της κατάσταση. Η μνήμη της είναι κενή και ο “fact-identifier” είναι στο 0. Εκτός από αυτό, αφαιρεί και όλους τους κανόνες που υπάρχουν. Ομοίως με την εντολή **(reset)** απομακρύνονται όλα τα γεγονότα από την “fact-list” και εισάγεται το αρχικό γεγονός με αριθμό ταυτότητας 0.

Για να απομακρυνθούν γεγονότα που έχουν εισαχθεί με την εντολή (deffacts), χρησιμοποιείται η εντολή **(undeffacts)**. Η διαφορά με την εντολή (retract) είναι ότι δεν απομακρύνεται ένα συγκεκριμένο γεγονός αλλά μια ομάδα γεγονότων. Για παράδειγμα στην ομάδα “important-people”, θα πληκτρολογηθεί (undeffacts important-people). Επίσης υπάρχει η ικανότητα να διαγραφτεί και το αρχικό γεγονός μέσω αυτής της εντολής.

2.3.2.3 Άλλες λειτουργίες της εντολής facts

Η εντολή **(facts)** δείχνει τα γεγονότα που βρίσκονται στη λίστα γεγονότων. Παρέχεται η δυνατότητα να τοποθετηθεί και ο “fact indentifier” δίπλα από την εντολή (facts) ώστε να φανερωθούν τα γεγονότα από τη τιμή αυτή και πάνω.

Παράδειγμα

(facts):δείχνει όλα τα γεγονότα.
(facts 3):δείχνει όλα τα γεγονότα από το γεγονός 3 και πάνω. Δηλαδή τα γεγονότα 0,1 και 2 δεν εμφανίζονται!

Η CLIPS παρέχει πολλές εντολές που βοηθούν στην αποσφαλμάτωση στα προγράμματα. Υπάρχει μια εντολή που επιτρέπει να παρατηρεί κανείς τα γεγονότα καθώς αυτά εισέρχονται και απομακρύνονται. Είναι πολύ βολικό από το να βάζει όλο την ίδια και την ίδια εντολή ξανά και ξανά για να δει τι γεγονότα υπάρχουν στη μνήμη. Για να το επιτύχει αυτό αρκεί να γράψει **(watch facts).** Εάν θελήσει να σταματήσει αυτή η εντολή αρκεί να γράψει κατά παρόμοιο τρόπο **(unwatch facts).** Μέσω της εντολής **(watch)** μπορεί να δει επιπλέον πράγματα όπως τα στιγμιότυπα (instances), τις ιδιότητες (slots), τους κανόνες, τα μηνύματα και άλλα. Η εντολή **(ppdeffacts)** δείχνει τα γεγονότα που έχουν δημιουργηθεί με την εντολή (deffacts) (Joseph C. Giarratano **2002**).

2.3.3 Τύποι πεδίων

Υπάρχουν διάφορα είδη πεδίων διαθέσιμα όπως τα πραγματικά (float), τα ακέραια (integer), αυτά που περιέχουν σύμβολα (symbol), οι συμβολοσειρές (string), οι εξωτερικές διευθύνσεις (external-adress), οι διευθύνσεις γεγονότων (fact-adress), τα ονόματα στιγμιότυπων (instance-name) και οι διευθύνσεις στιγμιότυπων (instance-adress). Ο τύπος κάθε

πεδίου καθορίζεται από την άξια που είναι αποθηκευμένη στο πεδίο. Σε ένα ανώνυμο πεδίο, ο τύπος καθορίζεται από το είδος του τύπου που θα εισαχθεί μέσα.

Ένα σύμβολο είναι ένας τύπος πεδίου που ξεκινάει με ένα τυπώσιμο ASCII χαρακτήρα και ακολουθείται επιλεκτικά από μηδέν έως περισσότερους τυπώσιμους χαρακτήρες. Τα πεδία είναι οροθετημένα ή δεμένα με μια ή περισσότερες παρενθέσεις και διαστήματα. Για παράδειγμα (duck-shot Brian Gary Rey) εδώ υπάρχουν τέσσερα πεδία που υποδεικνύουν ότι όλοι οι κυνηγοί σκοτώθηκαν από την πάπια δολοφόνο(!). Σε αυτό το γεγονός, τα πεδία καθορίζονται από τα διαστήματα καθώς και από το άνοιγμα και το κλείσιμο των παρενθέσεων (Joseph C. Giarratano **2002**).

Επίσης αξίζει να σημειωθεί ότι η CLIPS είναι ευαίσθητη σε κάποια σύμβολα όπως ΄΄,(,) ,& ,| ,< ,~ ,? ,; και $**.** Ωστόσο ενώ υπάρχει η δυνατότητα αυτά να χρησιμοποιηθούν, ίσως είναι πολύπλοκο για κάποιον να διαβάσει και να κατανοήσει τι κάνει ένα πρόγραμμα με την παρουσία τους. Γενικά μιλώντας, είναι καλό να αποφεύγεται να χρησιμοποιούνται αυτοί οι χαρακτήρες σε σύμβολα εκτός εάν υπάρχει κάποιος ιδιαίτερος λόγος. Σύμβολα μπορεί να είναι τα “george”, “george1”, “george23”, “george-g”, “george!&?”. Ο δεύτερος τύπος πεδίου είναι η συμβολοσειρά **(string)**. Ξεκινά και τελειώνει με διπλά εισαγωγικά. Κανένας έως πολλοί χαρακτήρες μπορούν να παρευρεθούν ανάμεσα στα εισαγωγικά. Για παράδειγμα “This is Sparta!”, ”excellent choice %^#$123 ”, ”may the force be with you”. Το τρίτο και τέταρτο είδος πεδίων είναι τα πεδία τιμών. Ένα πεδίο που μπορεί να αναπαριστά έναν αριθμό μπορεί να είναι είτε ακέραιος **(integer)** είτε πραγματικός **(float).** Όλοι οι αριθμοί στη CLIPS χαρακτηρίζονται ως ακέραιοι ή διπλής ακρίβειας πραγματικοί (Joseph C. Giarratano **2002**).

Είναι καλό να χρησιμοποιείται το πρώτο πεδίο ενός γεγονότος για να περιγράψει τη σχέση με τα υπόλοιπα πεδία. Εάν γίνει κάτι τέτοιο τότε εισάγεται η έννοια της σχέσης **(relation)**. Τα εναπομείναντα πεδία του γεγονότος χρησιμοποιούνται για συγκεκριμένες τιμές.

Παράδειγμα

(heroes-list spiderman cartain-america punisher hulk iron-man- x-men fantastic-four)

Κατά αυτόν τον τρόπο οι σχέσεις κάνουν περισσότερο αίσθηση σε ένα άτομο που δεν ασχολείται με τον προγραμματισμό και τον βοηθούν περισσότερο να κατανοήσει το πρόγραμμα από το να βλέπει απομονωμένες και σκόρπιες λέξεις που δεν έχουν καμία οπτική σχέση μεταξύ τους.

2.3.4 Κανόνες

Για να γίνει μια αξιόλογη δουλειά προγραμματίζοντας με τη CLIPS, ένα έμπειρο σύστημα πρέπει να έχει κανόνες όσο και γεγονότα. Ένας κανόνας είναι παρόμοιος με μια δήλωση «εάν …τότε…» σε μια διαδικαστική γλώσσα όπως η Ada, η Pascal ή η C. Η CLIPS όμως έχει το πλεονέκτημα ότι οι

κανόνες μπορούν να γραφτούν ή να φορτωθούν σε αυτήν μέσω ενός αρχείου κανόνων που μπορεί να δημιουργηθεί από έναν text editor (Joseph C. Giarratano **2002**).
Η γενική σύνταξη ενός κανόνα στην CLIPS s είναι η εξής:

```
(defrule     όνομα κανόνα     "σχόλιο"          ;επικεφαλίδα κανόνα
        συνθήκη-1                               ;μέρη συνθήκης
        …                                       ;LHS-left hand side
        συνθήκη-n
=>                                              ;βέλος που δηλώνει τι θα γίνει
μετά
        εντολή-1                                ;εντολή
        …                                       ;RHS-right hand side
        εντολή-m)
```

Η γενική σύνταξη μπορεί να ερμηνευτεί και με αυτόν τον τρόπο:

Αυτό που ισχύει
=>
Αυτό που θα συμβεί

Παράδειγμα:

```
(defrule george  "george puts always his name in rules!"
(great-is george)
=>
(assert (great-is his-heart)))
```

Το σχόλιο πρέπει να περικλείεται από εισαγωγικά ώστε να μην θεωρείται κάτι εκτελέσιμο από τη CLIPS και τοποθετείται πάντα πριν τη πρώτη συνθήκη του κανόνα και μετά το όνομα του. Ένας κανόνας μπορεί να έχει από καθόλου έως πολλές συνθήκες-οι οποίες μπορεί να αποτελούνται από ένα ή περισσότερα πεδία (η συνθήκη "great-is george" περιέχει τα πεδία "great-is" και "george") στο LHS ομοίως και με τις δράσεις στο RHS. Οι κανόνες μπορούν να γραφτούν και σε μια σειρά αλλά είναι προτιμότερο τα ξεχωριστά κομμάτια τους να γράφονται σε διαφορετικές σειρές ώστε να υπάρχει αξιοπιστία και λιγότερη οπτική πολυπλοκότητα. Γράφοντας τον κανόνα σωστά θα εμφανιστεί πάλι το λογότυπο της CLIPS στο παράθυρο. Κάνοντας όμως ένα οποιοδήποτε λάθος, εμφανίζεται ένα μήνυμα σφάλματος. Είναι σημαντικό να θυμάται κανείς ότι πρέπει να υπάρχει ίδιος αριθμός παρενθέσεων και στα αριστερά και στα δεξιά (Joseph C. Giarratano **2002**).

Αξιοσημείωτο είναι το γεγονός ότι εάν εισαχθεί κανόνας με το ίδιο όνομα τότε η CLIPS θα κρατήσει τον καινούργιο στην μνήμη και θα διαγράψει τον παλιό. Μόνο ένας κανόνας με ένα συγκεκριμένο όνομα μπορεί να υπάρχει στη CLIPS μια μόνο φορά. Αυτό συμβαίνει και σε άλλες προγραμματιστικές

γλώσσες στις οποίες ένα διαδικαστικό όνομα μπορεί να χρησιμοποιηθεί μόνο μια φορά για να αναγνωρίσει μια συγκεκριμένη διαδικασία.

Ολόκληρος ο κανόνας πρέπει να περικλείεται από παρενθέσεις. Κάθε συνθήκη και εντολή του πρέπει να περικλείονται και αυτές από παρενθέσεις. Μια εντολή είναι συνήθως μια λειτουργία που τυπικά δεν επιστρέφει τιμή αλλά αποδίδει κάποια χρήσιμη πράξη όπως η εισαγωγή ή η διαγραφή γεγονότων. Μια λειτουργία στην CLIPS είναι ένα κομμάτι ενός εκτελέσιμου κώδικα που αναγνωρίζεται από ένα συγκεκριμένο όνομα το οποίο επιστρέφει μια χρήσιμη τιμή και πραγματοποιεί ένα αποτέλεσμα (side-effect) όπως η εκτύπωση (Joseph C. Giarratano **2002**).

Η CLIPS επιχειρεί να ταιριάξει τις συνθήκες των κανόνων με τα γεγονότα που υπάρχουν στην λίστα γεγονότων. Εάν όλες οι συνθήκες ταιριάξουν, τότε πραγματοποιείται η ενεργοποίηση του κανόνα και η τοποθέτηση του στην ατζέντα. Όλοι λοιπόν οι κανόνες των οποίων οι συνθήκες ικανοποιούνται εισάγονται στην ατζέντα όπου υπάρχει ένα σύνολο συγκρούσεων (conflict set). Από την ατζέντα επιλέγεται κάθε φορά ένας μόνο κανόνας, ο οποίος και πυροδοτείται με βάση δύο κριτήρια (Ι. Βλαχάβας και συνεργάτες **2006**):

- τη προτεραιότητα των κανόνων
- τη στρατηγική επίλυσης συγκρούσεων

Η ατζέντα (agenda) είναι μια λίστα από ενεργοποιήσεις εκείνων των κανόνων όπου οι συνθήκες τους έχουν ταιριάξει με τα γεγονότα που υπάρχουν στην λίστα γεγονότων. Μπορεί να περιέχει από καμία έως πολλές ενεργοποιήσεις. Ένα πρόγραμμα θα σταματήσει την εκτέλεση του μόνο όταν δεν θα υπάρχουν ενεργοποιήσεις στην ατζέντα. Εάν υπάρχουν πολλές, η CLIPS αυτόματα καθορίζει ποια ενεργοποίηση θα γίνει πρώτη και τις διατάσει με βάση τον βαθμό προτεραιότητας που έχουν **(salience)**. Η ατζέντα συμπεριφέρεται σαν στοίβα (stack) όπου όσο μεγαλύτερη προτεραιότητα έχει ένας κανόνας τόσο πιο ψηλά βρίσκεται σε αυτή. Δηλαδή κάθε φορά εκτελείται ο κανόνας που βρίσκεται στην κορυφή της στοίβας. Ένας νέος κανόνας τοποθετείται στην ατζέντα σύμφωνα με τα ακόλουθα κριτήρια (Ι. Βλαχάβας και συνεργάτες **2006**):

- Προτεραιότητα (salience)
- Στρατηγική Επίλυσης Συγκρούσεων
- Αυθαίρετη σειρά

Οι νέοι κανόνες μπαίνουν "πάνω" από όλους τους κανόνες με μικρότερη ή ίση προτεραιότητα και "κάτω" από όλους τους κανόνες με μεγαλύτερη. Στους κανόνες με ίδια προτεραιότητα χρησιμοποιείται η τρέχουσα στρατηγική επίλυσης συγκρούσεων για να καθοριστεί η σειρά τους (Ι. Βλαχάβας και συνεργάτες **2006**). Με την εντολή **(agenda)** έχει κάποιος την δυνατότητα να δει τι ενεργοποιήσεις υπάρχουν στην ατζέντα. Η CLIPS εκτελεί τις εντολές που βρίσκονται στο RHS με υψηλότερη προτεραιότητα κανόνων στην ατζέντα πρώτα. Μετά ο κανόνας απομακρύνεται και εκτελούνται οι αμέσως επόμενες εντολές του νέου κανόνα με τον υψηλότερο βαθμό προτεραιότητας. Εάν κάποιοι κανόνες ενεργοποιήθηκαν από το ίδιο σύνολο

γεγονότων και τα προηγούμενα βήματα δεν μπόρεσαν να ορίσουν μία σειρά, τότε δίνεται σε αυτούς μια αυθαίρετη σειρά (όχι τυχαία), η οποία εξαρτάται από την υλοποίηση του συστήματος.

Παράδειγμα

CLIPS>(agenda)
0 george: f-0
for a total of 1 activation

Εδώ το 0 σημαίνει το βαθμό προτεραιότητας (salience) του “george” ενεργοποίησης και το f-0 είναι το “fact-identifier” του γεγονότος (great-is george) που ταιριάζει με την ενεργοποίηση. Η προτεραιότητα ενός κανόνα χαρακτηρίζεται από κάποιες ιδιότητες (Ι. Βλαχάβας και συνεργάτες **2006**):

- ✓ Είναι ακέραια αριθμητική τιμή.
- ✓ Όσο μεγαλύτερη είναι, τόσο μεγαλύτερη είναι και η προτεραιότητα του κανόνα
- ✓ Οι επιτρεπτές τιμές είναι από -10000 έως 10000
- ✓ Εάν δεν υπάρχει δήλωση, ο κανόνας θεωρείται ότι έχει την προκαθορισμένη τιμή μηδέν.

Στα έμπειρα συστήματα, η λέξη στρατηγική αναφέρεται στη τακτική αποφυγής συγκρούσεων μεταξύ διαφόρων ενεργοποιήσεων. Λογικό είναι κάποιος να αναρωτηθεί τι ρόλο παίζει η τακτική αυτή από την στιγμή που έχει την δυνατότητα να σχεδιάσει κατάλληλα το έμπειρο σύστημα ώστε να αποφύγει τις πιθανές συγκρούσεις. Και έχει δίκαιο. Εκεί όμως που πρέπει να δοθεί βαρύτητα είναι στο γεγονός ότι αυτή η εφαρμογή που έχει δημιουργήσει μπορεί να υλοποιηθεί με άλλες γλώσσες όποτε ποιος ο λόγος να μπαίνει στον κόπο να φτιάξει ένα έμπειρο σύστημα μέσω CLIPS; Η CLIPS παρέχει επτά διαφορετικές στρατηγικές επίλυσης συγκρούσεων όπως βάθους (depth), πλάτους (breadth), LEX, MEA, πολυπλοκότητας (complexity), απλότητας (simplicity) και τυχαιότητας (random). Στη στρατηγική βάθους οι νέες ενεργοποιήσεις που θα εισέλθουν στην ατζέντα, θα τοποθετηθούν πιο κάτω από εκείνες που έχουν υψηλότερο βαθμό προτεραιότητας αλλά πιο πάνω από αυτές που έχουν τον ίδιο ή χαμηλότερο. Γενικά δηλαδή οι ενεργοποιήσεις με αυτήν τη στρατηγική τοποθετούνται από αυτές που έχουν υψηλότερο βαθμό προς αυτές που έχουν χαμηλότερο. Στη στρατηγική πλάτους οι νέοι κανόνες μπαίνουν κάτω από τους παλιούς. Στη στρατηγική απλότητας οι κανόνες με πιο «απλές» συνθήκες κατατάσσονται πάνω από τους κανόνες με τις πιο πολύπλοκες σε αντίθεση με τη στρατηγική πολυπλοκότητας όπου κάνει το ακριβώς ανάποδο. Όσον αφορά τη στρατηγική LEX οι κανόνες που ενεργοποιούνται από "νεότερα" γεγονότα κατατάσσονται υψηλότερα στη ατζέντα. Για τους κανόνες που κατατάσσονται στην "ίδια ομάδα" με βάση το προηγούμενο κριτήριο, υψηλότερα κατατάσσονται εκείνοι που έχουν τις περισσότερες συνθήκες. Μπορεί να πει κανείς ότι η LEX αποτελεί συνδυασμό των στρατηγικών βάθους και πλάτους. Για τη MEA στρατηγική εξετάζεται το γεγονός το οποίο αντιστοιχεί στη πρώτη συνθήκη και οι κανόνες διατάσσονται με βάση το πότε εισήχθηκε αυτό στη λίστα. Όσο νεότερο είναι το γεγονός τόσο πιο "ψηλά" μπαίνει ο κανόνας. Για κανόνες οι οποίοι έχουν την ίδια σειρά

με βάση αυτό το κριτήριο χρησιμοποιείται η στρατηγική LEX ενώ στην στρατηγική τυχαιότητας οι κανόνες εισέρχονται με τυχαίο τρόπο στην ατζέντα (Ι. Βλαχάβας και συνεργάτες **2006**). Ωστόσο είναι πολύ δύσκολο να επιλέξει κανείς ποια είναι η καλύτερη στρατηγική ενώ μέρος της επιλογής διαδραματίζει και η εφαρμογή που θα θελήσει κάποιος να αναπτύξει. Η στρατηγική του βάθους είναι η εξορισμού στρατηγική που χρησιμοποιεί η CLIPS.

Για να τρέξει ένα πρόγραμμα αρκεί να πληκτρολογηθεί η εντολή **(run)**. Για να σωθεί ένας κανόνας ώστε να μην χρειαστεί ξανά η εισαγωγή του αργότερα αρκεί να γράψει κάποιος **(save "όνομα κανόνα.clp")** με "clp" να σημαίνουν τα αρχικά της CLIPS για να θυμίζει ότι πρόκειται για ένα πηγαίο αρχείο με κώδικα CLIPS. Να αναφερθεί ότι ενώ έχει τρέξει ένα κανόνα και οι ενεργοποιήσεις του στην ατζέντα έχουν πραγματοποιηθεί, δεν υπάρχει η δυνατότητα να γίνει πάλι κάτι τέτοιο παρά μόνο εάν γίνει ξανά εισαγωγή ή αφαίρεση των γεγονότων του.

Εάν κάποιος θελήσει να δει την περιγραφή ενός κανόνα τότε μπορεί να πατήσει την εντολή **(ppdefrule όνομα κανόνα).**

Παράδειγμα

```
(ppdefrule george)
(defrule MAIN::george
(great-is george)
=>
(assert (great-is his-heart)))
```

με το "main" να αναφέρεται στο "main module" που είναι αυτός ο κανόνας είναι από εξορισμού. Στην CLIPS μπορούν να οριστούν "modules" για να μπουν οι κανόνες σε αναλογία με τις δηλώσεις που ενδέχεται να βρίσκονται σε διαφορετικές διαδικασίες και λειτουργίες άλλων προγραμματιστικών γλωσσών. Η χρήση των "modules" διευκολύνει την δημιουργία των έμπειρων συστημάτων που έχουν πολλούς κανόνες από τη στιγμή που αυτά είναι ομαδοποιημένα με τις δίκες τους ατζέντες για κάθε "module". Εάν κάποιος έχει ξεχάσει το όνομα κανόνων τότε και για αυτό η CLIPS έχει λύση. Αρκεί να γράψει **(rules)** και έτσι εμφανίζεται μια λίστα κανόνων που υπάρχουν στη CLIPS. Με την λειτουργία **(printout)** εκτυπώνεται οτιδήποτε στην οθόνη αρκεί αυτό να περικλείεται σε εισαγωγικά και έχει δίπλα το γράμμα "t". Το "t" συμβολίζει την συσκευή στην οποία θα αποσταλεί το μήνυμα που επιθυμεί ο χρήστης το οποίο βρίσκεται μέσα σε εισαγωγικά. Η συσκευή μπορεί να είναι η οθόνη του υπολογιστή ή ένα αρχείο. Το "crlf" δηλώνει αλλαγή γραμμής.

Παράδειγμα

```
(defrule george  "again himself"
(great-is george)
=>
(printout t "But his heart is great too" crlf)).
```

Τώρα τρέχοντας αυτόν τον κανόνα με την εντολή (run) θα εμφανιστεί στην οθόνη το μήνυμα:

«But his heart is great too».

Για να διαγράψει κάποιος κανόνες μπορεί να χρησιμοποιήσει την εντολή **(undefrule** όνομα-κανόνα**).**

2.3.5 Άλλες εντολές της CLIPS

Μέχρι στιγμής πραγματοποιήθηκε εκτενής αναφορά σε γενικές αλλά σημαντικότατες εντολές πάνω στην δημιουργία έμπειρων συστημάτων με την βοήθεια της γλώσσας CLIPS. Ωστόσο τα έμπειρα συστήματα αποτελούνται από εκατοντάδες μέχρι χιλιάδες κανόνες. Όπως αναφέρθηκε πιο πριν, ένας κανόνας μπορεί να έχει περισσότερες από μία συνθήκες.

Παράδειγμα

(defrule take-a-walk
(status walking)
(walk-sign walk)
=>
(printout t "go and move forward my friend" crlf)).

Εάν πληκτρολογηθεί (run), στην οθόνη του υπολογιστή θα τυπωθεί το μήνυμα "go". Για να γίνει όμως κάτι τέτοιο θα πρέπει να υπάρχουν και οι δυο συνθήκες του κανόνα "take-a-walk". Εδώ υπάρχει ο περιορισμός «λογικό ΚΑΙ (AND) υποθετικό στοιχείο». Διότι εάν μια από τις δυο συνθήκες δεν ήταν ικανοποιημένη τότε ο κανόνας δεν θα εκτελούνταν (Joseph C. Giarratano **2002**).

Σημείωση για την εντολή reset.

Η **(reset)** εντολή κάνει τρία πράγματα συνοψίζοντας (Joseph C. Giarratano **2002**):

- ✓ Πραγματοποιεί εισαγωγή στη CLIPS του αρχικού γεγονότος.
- ✓ Αφαιρεί τα υπάρχοντα γεγονότα από τη λίστα γεγονότων που μπορούν να απομακρύνουν ενεργοποιημένους κανόνες από την ατζέντα.
- ✓ Εισάγει γεγονότα από δηλώσεις deffacts.

Όπως συμβαίνει και σε άλλες προγραμματιστικές γλώσσες, η CLIPS παρέχει την ικανότητα να θέτει κανείς σημεία σταματήματος εκτέλεσης τα λεγόμενα "**breakpoints**". Ένα "breakpoint" είναι μια ένδειξη για το ποτέ η CLIPS πρέπει να σταματήσει την εκτέλεση προκείμενου να δώσει βαρύτητα στην εκτέλεση ενός ειδικού κανόνα. Ένα "breakpoint" θέτεται από την εντολή **(set-break)**. Η **(remove-break)** το απομακρύνει ενώ με την εντολή **(show-breaks)** παρατηρεί κανείς μια λίστα που θα περιέχει όλους τους κανόνες που

διαθέτουν “break-sets” (Joseph C. Giarratano **2002**). Η δομή των παραπάνω εντολών συνοψίζεται ως εξής:

(set-break όνομα κανόνα)
(remove-break όνομα κανόνα)
(show-breaks)

2.3.5.1 Συνθήκες κανόνων με περιορισμούς

Εκτός από την απλή ταυτοποίηση συνθηκών με γεγονότα, διατίθεται η ικανότητα να εισαχθούν περιορισμοί στις συνθήκες χρησιμοποιώντας πιο εκφραστικές δομές όπως χρήση συγκεκριμένων συνδετικών ή και την εισαγωγή συνθηκών υπό μορφή συναρτήσεων (test conditions).

Τα συνδετικά μπορεί να είναι η λογική άρνηση “~”, η λογική διάζευξη “|” ή η λογική σύζευξη “&”. Το συνδετικό "&" χρησιμοποιείται συνήθως σε συνδυασμό με άλλους περιορισμούς (Ι. Βλαχάβας και συνεργάτες **2006**).

Παράδειγμα

(defrule days
(month Jan day ~mon)
(hour 12|13)
=>
(printout t “Day is not Monday, but it is noon” crlf))

Με την τοποθέτηση του συνδετικού λογικής άρνησης στο (month Jan day ~mon) διαπιστώνει κανείς ότι στη διάρκεια της ημέρας αναφέρεται παντού εκτός από την διάρκεια του πρωινού ενώ ταυτοποιείται και με γεγονότα της μορφής (month Jan day <σύμβολο>). Το <σύμβολο> ωστόσο δεν πρέπει να είναι “mon”. Αντίθετα με την χρήση του συνδετικού της λογικής διάζευξης στην ώρα (hour 12|13) παρατηρείται ταυτοποίηση είτε με (hour 12) ή με (hour 13) (Ι. Βλαχάβας και συνεργάτες **2006**).

Εκτός από το ταίριασμα με τις συνθήκες, ένας κανόνας μπορεί να αποσπάσει πληροφορίες και με άλλους τρόπους. Η CLIPS μπορεί να διαβάσει τη πληροφορία από το πληκτρολόγιο χρησιμοποιώντας τη συνάρτηση (**read**). Για καλύτερη κατανόηση θα δοθεί ένα παράδειγμα.

Παράδειγμα

CLIPS> (clear)
CLIPS> (defrule read-input
(initial-fact)
=>
(printout t "Name a primary color" crlf)
(assert (color (read))))
CLIPS>
(defrule check-input
?color <- (color ?color-read&red|yellow|blue)
=>
(retract ?color)

(printout t "Correct" crlf))
CLIPS> (reset)
CLIPS> (run)
Name a primary color
Red
Correct
CLIPS> (reset)
CLIPS> (run)
Name a primary color
green
CLIPS> ; No "correct"

Εδώ παρατηρείται ότι ο κανόνας έχει σχεδιαστεί ώστε να χρησιμοποιεί πληροφορία που εισέρχεται από το πληκτρολόγιο στο RHS μέρος του κανόνα και έτσι είναι βολικό να πυροδοτηθεί ο κανόνας με το αρχικό γεγονός. Η συνάρτηση (**read**) διαβάζει μόνο ένα πεδίο γι' αυτό εάν γραφτεί "primary colour is red", μόνο η πρώτη λέξη θα διαβαστεί. Εάν θελήσει κανείς να διαβαστεί όλο σαν ένα πεδίο αρκεί να βάλει διπλά εισαγωγικά. Ένας ακόμη περιορισμός της (**read**) είναι ότι δεν χρησιμοποιεί παρενθέσεις εκτός εάν βρίσκονται σε δίπλα εισαγωγικά. Η συνάρτηση (**read**) εισάγει ουσιαστικά ένα σύμβολο από το πληκτρολόγιο και συνήθως χρησιμοποιείται σε συνδυασμό με την εντολή (**bind),** για την ανάθεση τιμής σε μεταβλητή στις ενέργειες ενός κανόνα (Joseph C. Giarratano **2002**).

Παράδειγμα

```
(defrule get-user-answer
        (initial-fact)
        =>
        (printout t "What's your name: ")
        (bind ?name (read))
        (assert (user-name ?name))
)
```

Σε αυτήν την περίπτωση εάν εισαχθεί το αρχικό γεγονός και ύστερα τρεχθεί το πρόγραμμα θα εμφανιστεί στην οθόνη το μήνυμα " **What's your name:** ". Μέσω της συνάρτησης (read) γράφουμε το όνομα και στην συνεχεία αυτό διαβάζεται. Η δομή της εντολής (bind) είναι (**bind <μεταβλητή> <τιμή>)** όπου σε μια μεταβλητή ανατίθεται μια τιμή.

Παράδειγμα

```
(defrule rule1 "example rule"
        (oldcost ?oldcost)
        (newcost ?newcost)
        =>
        (bind ?total_cost (+ ?newcost ?oldcost))
        (assert (cost ?total_cost))
        (printout t "The total cost is " ?total_cost crlf)
)
```

Σε αυτό το σημείο οι τιμές του “newcost” και του “oldcost” αναθέτονται σε μια νέα μεταβλητή που ονομάζεται “totalcost” με την βοήθεια της (bind) (Ι. Βλαχάβας και συνεργάτες **2006**). Η συνάρτηση **(readline)** χρησιμοποιείται για να διαβάζει πολλαπλές τιμές μέχρι τερματίζουν όλες. Αυτή η συνάρτηση διαβάζει τα δεδομένα ως συμβολοσειρές. Για να εισαχθούν τα δεδομένα της (readline) αρκεί να χρησιμοποιηθεί η συνάρτηση **(assert-string).**

Παράδειγμα

```
CLIPS> (clear)
CLIPS> (assert-string "(primary color is red)")
<Fact-0>
CLIPS> (facts)
f-0 (primary color is red)
For a total of 1 fact.
CLIPS>
```

Μια άλλη πολύ χρήσιμη εντολή είναι η **(test).** Μέσω αυτής της εντολής αντικαθιστώνται πιθανά “if” και “then” που μπορεί να υπάρχουν μέσα στο κανόνα. Παρακάτω διακρίνεται ένα παράδειγμα κανόνων για την ηλικία που περιέχει “if” και “then” (Ι. Βλαχάβας και συνεργάτες **2006**).

Παράδειγμα

Παλιός κανόνας

```
(defrule middle
        (age ?x)
   =>
        (if (and (> ?x 14) (< ?x 35))
        then (assert (agegroup middle))))
```

Νέος κανόνας

```
        (defrule middle "middle age rule"
        (age ?x)
        (test (and (> ?x 14) (< ?x 35)))
   =>
         (assert (agegroup middle)))
```

Η προσθήκη σχολίων κατά μήκος του κώδικα είναι αποδεκτή αρκεί μπροστά από το σχόλιο(α) να βρίσκεται ένα ερωτηματικό (;). Έτσι κατά συνέπεια η CLIPS δεν λαμβάνει υπόψη τα σχόλια ως εκτελέσιμο κώδικα.

2.3.6 Μεταβλητές

Όπως και σε άλλες προγραμματιστικές γλώσσες, έτσι και η CLIPS διαθέτει μεταβλητές για την αποθήκευση τιμών. Αντίθετα με ένα γεγονός που είναι στατικό, τα περιεχόμενα μιας μεταβλητής είναι δυναμικά και οι τιμές που της αναθέτονται αλλάζουν. Όταν ένα γεγονός εισέλθει, τα πεδία του αλλάζουν μόνο όταν αυτό διαγραφεί ή όταν δημιουργηθεί ένα καινούργιο με τα αλλαγμένα πεδία. Για την δήλωση μιας μεταβλητής αρκεί να γραφτεί μπροστά από γράμματα ή αριθμούς το σύμβολο (?) (Joseph C. Giarratano **2002**).

Παράδειγμα

?age, ?45, ?8takis

Πριν μια μεταβλητή χρησιμοποιηθεί πρέπει να της ανατεθεί μια τιμή. Εάν δεν πραγματοποιηθεί κάτι τέτοιο τότε η CLIPS θα ανταποκριθεί με εμφάνιση μηνύματος λάθους. Η μεταβλητή πρέπει να είναι δεμένη (bound) μέσα σε ένα κανόνα.

Παράδειγμα

(defrule george
(george-is ?x)
=>
(assert(he-is ?x)))

Σε αυτό το σημείο εάν εισαχθεί το γεγονός “george-is hansome” και ύστερα πληκτρολογηθεί (run), θα εισαχθεί μέσα στην CLIPS το γεγονός “he-is handsome”. Έτσι διαπιστώνεται ότι μια κοινή χρήση των μεταβλητών είναι το ταίριασμα μιας τιμής στο LHS μέρος ενός κανόνα και την εισαγωγή της μέσω δεσίματος (bound) στο RHS μέρος ενός κανόνα. Μια μεταβλητή μπορεί να χρησιμοποιηθεί πάνω από μια φορά. Μια άλλη λειτουργία των μεταβλητών είναι ότι χρησιμεύουν στην εκτύπωση μηνυμάτων στην οθόνη ενώ πάνω από μια μεταβλητές μπορούν να χρησιμοποιηθούν σε μια συνθήκη ενός κανόνα.

Η απομάκρυνση (retraction) είναι πολύ χρήσιμη στα έμπειρα συστήματα και συνήθως γίνεται στο RHS μέρος του κανόνα. Πριν ένα γεγονός διαγραφτεί, πρέπει να καθοριστεί στην CLIPS. Για να απομακρυνθεί ένα γεγονός από έναν κανόνα, η διεύθυνση του πρέπει πρώτα να δεθεί με μια μεταβλητή στο LHS του κανόνα.

Παράδειγμα

(defrule get-married
?person<-(bachelor George)
=>
(printout t “George is now living happily being married” ?person crlf)
(retract ?person))

Εάν πληκτρολογηθεί η εντολή (run) τότε εμφανίζεται “George is now living happily being married” <Fact-0> , με τη CLIPS να τυπώνει τον αριθμό ταυτότητας του “?person” αφού το βέλος δένει ή δείχνει τη διεύθυνση του γεγονότος στο “?person”. Επιπλέον δεν υπάρχει γεγονός “bachelor George” αφού έχει απομακρυνθεί. Επίσης μέσω των μεταβλητών μπορεί να βρεθεί μια τιμή γεγονότος ή μια διεύθυνση (Joseph C. Giarratano **2002**).

Παράδειγμα

```
(defrule marriage
?duck <- (bachelor ?name)
=>
(printout t ?name " is now happily married" crlf)
(retract ?duck))
CLIPS> (deffacts good-prospects
(bachelor George)
(bachelor Dorky)
(bachelor Dicky))
CLIPS> (reset)
CLIPS> (run)
Dicky is now happily married
Dorky is now happily married
George is now happily married
```

Όσα γεγονότα ταίριαξαν με το πρότυπο (bachelor ?name), εμφανίστηκαν.

Αντί να δένεται μια τιμή πεδίου σε μια μεταβλητή, η παρουσία ενός μη κενού πεδίου μπορεί να ανιχνευθεί χρησιμοποιώντας μια “wildcard”. Για παράδειγμα ένα άτομο θέλει να βγαίνει μόνο με άτομα όπου το πρώτο όνομα τους είναι “George”. Η πιο απλή μορφή μιας “wildcard” καλείται “single-field wildcard” και συμβολίζεται με ένα (?). Το (?) καλείται “single-field constraint”. Μια “single-field wildcard” ισχύει για ακριβώς ένα πεδίο (Joseph C. Giarratano **2002**).

Παράδειγμα

```
(defrule dating-men
(bachelor George ?)
=>
(printout t "Date George " crlf))
CLIPS>(deffacts duck
(bachelor Dicky)
(bachelor George)
(bachelor George Mallard)
(bachelor Dinky George)
(bachelor George Dinky Mallard))
CLIPS> (reset)
CLIPS> (run)
Date George
```

Εδώ παρατηρείται ότι μέσω του (bachelor George ?) το δεύτερο όνομα οποιουδήποτε “George” δεν έχει σημασία. Αρκεί το πρώτο όνομα να είναι “George” και κατά συνέπεια ο κανόνας ικανοποιείται και εκτελείται. Αντί όμως να γράφονται ξεχωριστοί κανόνες για το κάθε πεδίο, είναι πολύ πιο εύκολο να χρησιμοποιηθεί η “multifield wildcard” η λεγόμενη «πολλαπλών τιμών αγριόκαρτα». Αυτή συμβολίζεται με ($?)» και αναπαριστά από κανένα έως πολλά πεδία. Στο παραπάνω παράδειγμα εάν υπήρχε (bachelor George $?) τότε η CLIPS θα εκτύπωνε όλους αυτούς που έχουν πρώτο όνομα “George” (Joseph C. Giarratano **2002**).

2.3.7 Πρότυπα γεγονότων

Σε μεγάλα προγράμματα χρειάζεται να αναπαρασταθεί η πληροφορία με μεγάλα ή σύνθετα γεγονότα. Η CLIPS προσφέρει τα πρότυπα γεγονότων **(templates)** ως εναλλακτικό τρόπο δημιουργίας και διαχείρισης αυτών. Τα πρότυπα είναι μια δομή με την οποία μπορεί να οριστεί η μορφή που θα έχουν τα γεγονότα σε ένα πρόγραμμα και κάθε πρότυπο έχει ένα σύνολο από ιδιότητες **(slots),** στις οποίες μπορούν να ανατεθούν τιμές αυτόνομα (Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου **2006**). Το **(deftemplate)** καθορίζει μία ομάδα από συσχετισμένα μεταξύ τους πεδία με ένα σχέδιο παρόμοιο με τον τρόπο που ένα αρχείο Pascal είναι μια ομάδα από συσχετισμένα μεταξύ τους δεδομένα. Πιο αναλυτικά, ένα (deftemplate) είναι μια λίστα από ονομασμένα πεδία που καλούνται ιδιότητες. Επιτρέπει πρόσβαση κατά όνομα και συμβάλλει σε καλό στυλ σε προγράμματα έμπειρων συστημάτων ενώ παράλληλα είναι ένα πολύτιμο εργαλείο για τεχνολογία λογισμικού. Μία ιδιότητα λέγεται “single-slot” ή “multislot”. Το “single-slot” περιέχει μόνο ένα πεδίο ενώ το “multislot” περιέχει από κανένα μέχρι πολλά. Σε ένα (deftemplate) μπορούν να χρησιμοποιηθούν και τα “single-slot” και τα “multislot” σε οποιοδήποτε αριθμούς. Στις συνθήκες των κανόνων δε χρειάζεται να γραφεί ολόκληρο το γεγονός, αλλά μόνο το όνομα του προτύπου και τα ονόματα των ιδιοτήτων που ενδιαφέρουν. Για να καθοριστεί το επιτρεπτό εύρος τιμών που επιδέχεται μια ιδιότητα με αριθμητικές τιμές χρησιμοποιείται η εξής δήλωση:

(range <min value> <max value>)

ενώ για τις ιδιότητες με τις πολλαπλές τιμές

(cardinality <min> <max>).

Ακόμη αξιοσημείωτο είναι το γεγονός ότι εάν η ιδιότητα δεν πρέπει να έχει άνω ή κάτω όριο τότε εισάγεται το σύμβολο “?VARIABLE” στην αντίστοιχη θέση.

Παράδειγμα

(deftemplate prospect ;όνομα του deftemplate

"Vital information" ;σχόλιο
(slot name ;όνομα πεδίου
(type STRING) ;τύπος πεδίου
(default ?DERIVE)) ;default τιμή του πεδίου όνομα
(slot assets ;όνομα πεδίου
(type SYMBOL) ;τύπος πεδίου
(default rich)) ;default τιμή του πεδίου περ.στοιχεία
(slot age ;όνομα πεδίου
(type NUMBER) ;τύπος πεδίου-NUMBER(INTEGER ή FLOAT)
(default 80))) ;default τιμή του πεδίου ηλικία

Αυτό το (deftemplate) έχει τρεις ιδιότητες “single-slots” που καλούνται «όνομα», «ηλικία» και «περιουσιακά στοιχεία». Εάν εισαχθεί ένα ενδεχόμενο (prospect) χωρίς να του δώσουμε τιμές όπως (assert (prospect)) τότε η CLIPS θα βγάλει το εξής αποτέλεσμα (Joseph C. Giarratano **2002**):

(prospect (name "") (assets rich) (age 80))

δηλαδή θα έχει βάλει στις τρεις ιδιότητες “single-slot”, τις τιμές default που έχουν οριστεί στο (deftemplate prospect). Να σημειωθεί ότι εάν εισαχθεί άλλο “prospect” και δεν έχει οριστεί τιμή σε ένα από τα οποιοδήποτε “single-slot” τότε και εκεί η CLIPS θα καθορίσει και θα βάλει την εξορισμού τιμή. Η σειρά των “single-slot” που πληκτρολογούνται για να εισάχθει ένα ενδεχόμενο δεν διαδραματίζει κάποιον ρόλο. Η γενική δομή ενός (deftemplate) με N ιδιότητες είναι:

(deftemplate <name>
(slot-1)
(slot-2)
...
(slot-N)).

Οι ιδιότητες μπορούν να είναι διάφοροι τύποι όπως συνοψίζεται στο Πίνακα 2.3.7.1:

Πίνακας 2.3.7.1 Τύποι ιδιοτήτων που μπορεί να έχει ένα πρότυπο γεγονότων **(Ι. Βλαχάβας και συνεργάτες** 2006**)**

Τύπος	Η ιδιότητα μπορεί να περιέχει
SYMBOL	σύμβολα
STRING	αλφαριθμητικά
LEXEME	σύμβολα ή αλφαριθμητικά
INTEGER	ακέραιες τιμές
FLOAT	πραγματικές τιμές
NUMBER	ακέραιες ή πραγματικές τιμές
?VARIABLE	τιμές οποιουδήποτε τύπου

Η CLIPS είναι μια βασισμένη σε κανόνες γλώσσας που χρησιμοποιεί έναν πολύ αποτελεσματικό αλγόριθμο ταίριαξης συνθηκών που καλείται *Rete* Αλγόριθμος. Ο όρος Rete σημαίνει στα λατινικά δίκτυο και περιγράφει την αρχιτεκτονική του λογισμικού για τη διαδικασία ταίριαξης συνθηκών. Είναι όμως πολύ δύσκολο να δώσουμε ακριβής κανόνες που πάντα θα βελτιώνουν τη αποτελεσματικότητα ενός προγράμματος που θα τρέχει με τον αλγόριθμο *Rete*. Ωστόσο υπάρχουν κάποιοι γενικοί και ανεπίσημοι κανόνες που μπορούν να βοηθήσουν (Joseph C. Giarratano **2002**):

- ✓ Οι πιο συγκεκριμένες συνθήκες πρέπει να μπαίνουν σε ένα κανόνα πρώτες. Οι συνθήκες με τις αδέσμευτες μεταβλητές και τις “wildcards” πρέπει να είναι πιο κάτω στην λίστα με τις συνθήκες του κανόνα.
- ✓ Οι συνθήκες με λιγότερα “matching” γεγονότα θα πρέπει να είναι οι πρώτες που θα ελαχιστοποιούν τα μερικά ταιριάσματα (partial matches).
- ✓ Συνθήκες που συχνά εισέρχονται και αποσύρονται (volatile), πρέπει να τοποθετούνται στο τέλος της λίστας των συνθηκών.

Test conditional element: πραγματοποιείται σύγκριση αριθμών, μεταβλητών και συμβολοσειρών στο LHS μέρος ενός κανόνα. Χρησιμοποιείται ως μια συνθήκη στο LHS. Ένας κανόνας θα εκτελεστεί μόνο εάν το (test) ικανοποιηθεί μαζί με τις άλλες συνθήκες. Οι συναρτήσεις που μπορούν να χρησιμοποιηθούν με το “test conditional element” διακρίνονται στους Πίνακες 2.3.7.2, 2.3.7.3 και 2.3.7.4. και χωρίζονται σε λογικές, αριθμητικές και σύγκρισης.

Πίνακας 2.3.7.2 Λογικές συναρτήσεις που μπορούν να χρησιμοποιηθούν με το test conditional element (Joseph C. Giarratano **2002**)

	Λογικές
not	Boolean not
and	Boolean and
or	Boolean or

Πίνακας 2.3.7.3 Αριθμητικές συναρτήσεις που μπορούν να χρησιμοποιηθούν με το test conditional element (Joseph C. Giarratano **2002**)

	Αριθμητικές
+	Πρόσθεση
-	Αφαίρεση
*	Πολλαπλασιασμός
/	Διαίρεση

Πίνακας 2.3.7.4 Συναρτήσεις σύγκρισης που μπορούν να χρησιμοποιηθούν με το test conditional element (Joseph C. Giarratano **2002**)

	Σύγκρισης
eq	Συγκρίνει τύπο και πλάτος. Για όλους τους τύπους
neq	Όχι ίσο. Για όλους τους τύπους
=	Ίσο-για αριθμητικούς τύπους. Συγκρίνει πλάτος
<>	Όχι ίσο. Για αριθμητικούς τύπους
>=	Μεγαλύτερο ή ίσο από
>	Μεγαλύτερο από
<=	Μικρότερο ή ίσο από
<	Μικρότερο από

Όλες οι συναρτήσεις σύγκρισης θα εμφανίσουν λάθος μήνυμα εκτός από την “eq” και την ”neq” εάν συγκρίνουν έναν αριθμό και έναν μη αριθμό. Εάν δεν είναι γνωστός ο τύπος, τότε αυτές οι δυο είναι προτιμότερο να χρησιμοποιούνται. Η πρώτη ελέγχει για το αν υπάρχει το ίδιο πλάτος και ίδιος τύπος ενώ η “=” συνάρτηση ελέγχει μόνο το πλάτος των αριθμητικών arguments και δεν ενδιαφέρεται εάν είναι ακέραιοι ή πραγματικοί (Joseph C. Giarratano **2002**).

Οι λογικές συναρτήσεις είναι όπως είδαμε και στον παραπάνω πίνακα, οι “not”, “or” και “and”. Η CLIPS παρέχει τη δυνατότητα να συγκεκριμενοποιούμε ένα «ακραίο και υποθετικό στοιχείο (explicit and conditional element)» ή ένα «ή υποθετικό στοιχείο (or conditional element)» ή και ένα «όχι υποθετικό στοιχείο (not conditional)» στο LHS μέρος ενός κανόνα. Το «υποθετικό λογικό στοιχείο» χρησιμοποιεί την λέξη κλειδί “logical” σε μια συνθήκη για να υποδείξει ότι οι οντότητες που ταιριάζουν με τις συνθήκες παρέχουν λογική υποστήριξη στις εισαγωγές στο RHS μέρος του κανόνα. Παρόλο που η λογική υποστήριξη δουλεύει για τις εισαγωγές, δεν χρησιμοποιείται στις εισαγωγές απομακρυσμένων γεγονότων. Το ηθικό

δίδαγμα σε αυτή την περίπτωση είναι ότι εάν χαθεί κάτι λόγω του τεράστιου όγκου της πληροφορίας δεν μπορεί να ανακτηθεί πίσω (Joseph C. Giarratano **2002**).

Εκτός από το συνδετικό περιορισμό που χρησιμοποιεί τα “~”, ”|” και ”& ” υπάρχει και ένας άλλος τύπος περιορισμού τύπου που λέγεται κατηγορηματικός περιορισμός (predicate constraint). Χρησιμοποιείται για ταίριασμα γεγονότων πιο πολύπλοκων πεδίων. Ο σκοπός του είναι ο περιορισμός ενός πεδίου που εξαρτάται από το αποτέλεσμα μιας “Boolean” έκφρασης. Εάν η “Boolean” επιστρέψει “FALSE”, ο περιορισμός δεν είναι ικανοποιητικός και έτσι το ταίριασμα αποτυγχάνει. Ο κατηγορηματικός περιορισμός είναι πολύ χρήσιμος ιδιαίτερα με αριθμητικές συνθήκες (Joseph C. Giarratano **2002**).

Μια κατηγορηματική συνάρτηση είναι αυτή που επιστρέφει “FALSE” ή “non-FALSE” τιμή. Το “:” που ακολουθείται από μια κατηγορηματική συνάρτηση καλείται κατηγορηματικός περιορισμός. Το ”:” μπορεί να διαδέχεται από “&”, “|”, ή “~” ή μπορεί να στέκεται και μόνο του μέσα σε μια συνθήκη. Τυπικά χρησιμοποιείται μαζί με το “&” συνδετικό περιορισμό και συμβολίζεται έτσι: “**&:**” (Joseph C. Giarratano **2002**). Στον Πίνακα 2.3.7.5 παρουσιάζονται πιο αναλυτικά οι κατηγορηματικές λειτουργίες.

Πίνακας 2.3.7.5 Κατηγορηματικές λειτουργίες (Joseph C. Giarratano **2002**)

Κατηγορηματικές Λειτουργίες	**Check if <arg> is**
(**evenp** <arg>)	evenp αριθμός
(**floatp** <arg>)	Πραγματικός αριθμός
(**integerp** <arg>)	Ακέραιος
(**lexemep** <arg>)	Σύμβολο ή συμβολοσειρά
(**numberp** <arg>)	Πραγματικός ή ακέραιος
(**oddp** <arg>)	Odd αριθμός
(**pointerp** <arg>)	Εξωτερική διεύθυνση
(**sequencep** <arg>)	Τιμή με πολλαπλά πεδία
(**stringp** <arg>)	Συμβολοσειρά
(**symbolp** <arg>)	Σύμβολο

Η CLIPS παρέχει επίσης και κάποιες διαδικαστικές προγραμματιστικές δομές που μπορούν να χρησιμοποιηθούν στο RHS μέρος των κανόνων. Αυτές είναι οι “while” και “if then else” που βρίσκονται επίσης και σε άλλες γλώσσες όπως η Ada, C και Pascal. Άλλη χρήσιμη λειτουργία για τους βρόγχους οι οποίοι μπορούν να προκύψουν με τη χρήση της δομής “while”, είναι η “break”. Με αυτή τερματίζουμε τη τρέχουσα εκτέλεση του “while” βρόγχου. Η λειτουργία “return” αμέσως τελειώνει την τρέχουσα εκτελούμενη “deffunction”, “generic function”, “method” ή “message handler”. Κλείνοντας να αναφερθεί μια τελευταία λειτουργία που ονομάζεται «λειτουργία εκτίμησης» δηλαδή “eval”. Χρησιμοποιείται για την εκτίμηση οποιονδήποτε συμβολοσειρών ή συμβολών εκτός από τις “def” τύπου δομές όπως “defrule”, “deffacts” κλπ (Joseph C. Giarratano **2002**).

2.3.8 Αντικειμενοστραφής γλώσσα προγραμματισμού COOL

Η αντικειμενοστραφής γλώσσα προγραμματισμού της CLIPS (CLIPS Object Oriented Language) χαρακτηρίζεται από πέντε βασικά στοιχεία: την **αφαίρεση** (abstraction), την **ενθυλάκωση** (encapsulation), τη **κληρονομικότητα** (inheritance), τον **πολυμορφισμό** (polymorphism) και τη **δυναμική δέσμευση** (dynamic binding).

Στη COOL μια κλάση είναι ένα πρότυπο που περιγράφει τα κοινά χαρακτηριστικά ή τα γνωρίσματα των αντικειμένων. Εδώ η λέξη πρότυπο χρησιμοποιείται με τη λογική ενός εργαλείου που εφαρμόζεται για τη κατασκευή αντικειμένων που διαθέτουν κοινά γνωρίσματα. Οι κλάσεις των αντικειμένων ταξινομούνται σε μια ιεραρχία ή ένα γράφημα ώστε να γίνεται πλήρη περιγραφή των σχέσεων ανάμεσα στα αντικείμενα. Κάθε κλάση είναι μια αφαίρεση ενός συστήματος πραγματικού κόσμου ή κάποιου αλλού λογικού συστήματος που προσπαθούμε να μοντελοποιήσουμε. Ο όρος «αφαίρεση» αναφέρεται (Joseph C. Giarratano **2002**):

- ✓ Στην αφαιρετική περιγραφή ενός αντικειμένου πραγματικού κόσμου ή ενός συστήματος το οποίο θα μοντελοποιηθεί ή
- ✓ Στη διαδικασία της αναπαράστασης ενός συστήματος με τις έννοιες των κλάσεων.

2.3.8.1 Αφαίρεση

Η αφαίρεση είναι ένα από τα πέντε γενικώς αποδεκτά χαρακτηριστικά της COOL γλώσσας. Τα υπόλοιπα είναι η κληρονομικότητα, η ενθυλάκωση, ο πολυμορφισμός και η δυναμική συνεκτικότητα. Ο όρος **«αφηρημένη»** (abstruct) σημαίνει ότι δεν απασχολούν οι ασήμαντες και οι μη απαραίτητες λεπτομέρειες. Μια αφηρημένη περιγραφή ενός πολύπλοκου συστήματος είναι μια απλοποιημένη περιγραφή που επικεντρώνεται στη σχετική πληροφορία για ένα συγκεκριμένο σκοπό. Ως συνέπεια το σύστημα αναπαριστάται από ένα πιο εύκολο στην κατανόηση μοντέλο. Ένα παράδειγμα είναι η οδήγηση των ανθρώπων. Δεν χρειάζεται να τους απασχολούν άλλα χίλια δυο πράγματα όταν οδηγούν. Το μυαλό τους το έχουν επικεντρωμένο στο τιμόνι, στο κιβώτιο ταχυτήτων και στα πετάλια (Joseph C. Giarratano, **2002**). Με τη βοήθεια της αφαίρεσης ο ορισμός νέων κλάσεων επιτρέπει την αφαίρεση νέων τύπων δεδομένων. Οι ιδιότητες (slots) και οι μέθοδοι (message-handlers) αυτών των κλάσεων περιγράφουν τις ιδιότητες (properties) και τη συμπεριφορά (behavior) μίας καινούριας ομάδας αντικειμένων (Ι. Βλαχάβας και συνεργάτες **2006**).

2.3.8.2 Πολυμορφισμός

Ο πολυμορφισμός είναι ένα από τα πέντε όπως αναφέρθηκε πιο πάνω στοιχεία που χαρακτηρίζουν την γλώσσα COOL. Πολυμορφισμός είναι η ιδιότητα ενός αντικειμένου όπου μπορεί να απαντήσει σε ένα μήνυμα με ένα

εντελώς διαφορετικό τρόπο από ότι ένα άλλο αντικείμενο. Ο πολυμορφισμός επιτυγχάνεται επισυνάπτοντας μεθόδους, που έχουν το ίδιο όνομα αλλά εκτελούν διαφορετικές ενέργειες, στις κλάσεις των δύο αντικειμένων αντίστοιχα (Ι. Βλαχάβας και συνεργάτες **2006**).

2.3.8.3 Δυναμική δέσμευση

Η ιδιότητα της δυναμικής δέσμευσης υποστηρίζεται στη COOL με την έννοια ότι μια αναφορά αντικειμένου (object reference) κατά τη κλήση μίας συνάρτησης δεν έχει τιμή μέχρι τη στιγμή της εκτέλεσης. Για παράδειγμα, ένα όνομα στιγμιότυπου (instance-name) ή μια μεταβλητή μπορεί να αναφέρεται σε ένα αντικείμενο όταν στέλνεται ένα μήνυμα και να αναφέρεται σε ένα άλλο αντικείμενο κάποια άλλη στιγμή αργότερα (Ι. Βλαχάβας και συνεργάτες **2006**).

2.3.8.4 Κληρονομικότητα

Οι κλάσεις είναι ταξινομημένες σε μια ιεραρχία με τις πιο γενικές να βρίσκονται στη κορυφή και τις πιο εξειδικευμένες να είναι πιο κάτω. Οι κλάσεις που βρίσκονται ψηλά στην ιεραρχία έχουν κάποια γενικά χαρακτηριστικά και μεθόδους τα οποία είναι κοινά για όλες τις κλάσεις που βρίσκονται χαμηλότερα στην ιεραρχία. Με αυτόν τον τρόπο είναι πιο εύκολο ο ορισμός νέων κλάσεων ως εξειδικευμένες βελτιστοποιήσεις ή ως τροποποιήσεις των ήδη υπαρχόντων. Η χρήση της κληρονομικότητας μπορεί να επιταχύνει την ανάπτυξη λογισμικού και την αύξηση της αξιοπιστίας επειδή το νέο λογισμικού δεν χρειάζεται να δημιουργείται από το μηδέν κάθε φορά που ένα καινούργιο πρόγραμμα σχεδιάζεται αφού αποφεύγεται η επανάληψη του ορισμού κοινών χαρακτηριστικών και μεθόδων (Joseph C. Giarratano, **2002,** Ι. Βλαχάβας και συνεργάτες **2006**). Κατά συνέπεια η δομή και η συμπεριφορά μιας γενικότερης κλάσης κληρονομείται στις περισσότερο συγκεκριμένες.

Με σκοπό τον καθορισμό μιας κλάσης, πρέπει να ορίσουμε μια ή περισσότερες κλάσεις-γονείς ή υπερκλάσεις της κλάσης που θα οριστεί. Το αντίθετο μιας υπερκλάσης είναι η κλάση-παιδί ή υποκλάση. Αυτό καθορίζει τη κληρονομικότητα της νέας κλάσης. Μια υποκλάση κληρονομεί τα γνωρίσματα από μια ή περισσότερες υποκλάσεις.

Πολλαπλή κληρονομικότητα

Μία κλάση μπορεί να συνδέεται ιεραρχικά με περισσότερες από μία γενικότερες κλάσεις. Η κλάση που συνδέεται με πολλαπλές γενικότερες κλάσεις κληρονομεί χαρακτηριστικά και μεθόδους από όλες. Η COOL, χρησιμοποιώντας την υπάρχουσα ιεραρχία των κλάσεων, ορίζει τη **λίστα προτεραιότητας κλάσεων** (class precedence list) για μία νέα κλάση. Αντικείμενα, τα οποία είναι στιγμιότυπα της νέας κλάσης, μπορούν να κληρονομήσουν ιδιότητες και συμπεριφορά από κάθε μία από τις κλάσεις της

λίστας αυτής. Η λέξη προτεραιότητα υποδηλώνει ότι μία μέθοδος μίας κλάσης, που είναι πρώτη στη λίστα, υπερισχύει της ίδιας μεθόδου άλλης κλάσης που βρίσκεται πιο μετά στη λίστα (Ι. Βλαχάβας και συνεργάτες **2006**).

Ο όρος γνώρισμα στο COOL αναφέρεται στις ιδιότητες ενός αντικειμένου οι λεγόμενες **slots** που το περιγράφουν. Για παράδειγμα ένα αντικείμενο για την αναπαράσταση ενός σούπερ ήρωα έχει ιδιότητες για το όνομα ήρωα, κανονικό όνομα, επίθετο, δυνάμεις, νέμεση του ήρωα. Ένα **στιγμιότυπο** (instance) είναι ένα αντικείμενο που έχει τιμές για τις ιδιότητες όπως “Batman”, “Bruce”, “Wayne”, “Genius”, “Joker”. Οι πιο χαμηλές σε επίπεδο κλάσεις κληρονομούν τις ιδιότητες τους από τις κλάσεις που βρίσκονται σε ανώτερο επίπεδο. Οι νέες ιδιότητες καθορίζονται επιπρόσθετα με τις κληρονομημένες ιδιότητες ώστε να δημιουργηθούν όλα τα χαρακτηριστικά που θα περιγράφουν την νέα κλάση.

Η συμπεριφορά ενός αντικειμένου ορίζεται από τους χείριστες μηνυμάτων **(message-handlers).** Ένας χείριστης μηνύματος ενός αντικειμένου άπαντα στα μηνύματα και εκτελεί τις αιτούμενες πράξεις.

Παράδειγμα

(send [Batman] print)

Σε αυτό το σημείο ο κατάλληλος χείριστης μηνύματος θα εκτυπώσει τις τιμές των ιδιοτήτων του στιγμιότυπου “Batman”.

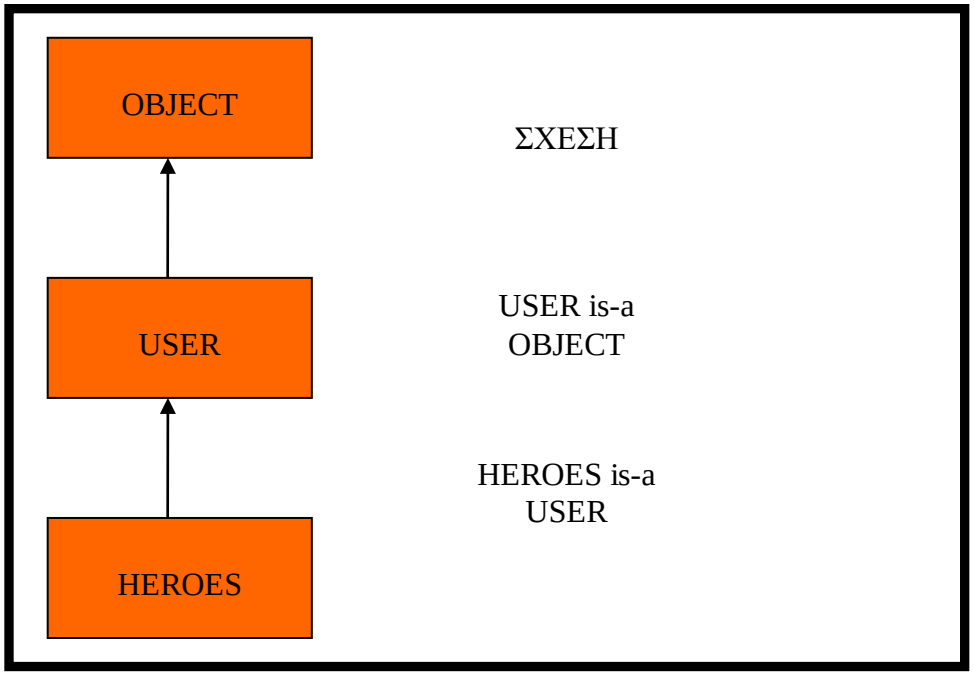
Γενικά τα στιγμιότυπα ορίζονται μέσα σε [ ]. Ένα μήνυμα ξεκινά με την λειτουργία “send” και ακολουθείται αργότερα από το όνομα του στιγμιότυπου, το όνομα του μηνύματος και τα όποια απαιτούμενα ορίσματα. Ένα αντικείμενο στην CLIPS είναι ένα στιγμιότυπο σε μια κλάση (Joseph C. Giarratano **2002**).

2.3.8.5 Ενθυλάκωση

Ο όρος «ενθυλάκωση» σημαίνει ότι μια κλάση καθορίζεται σε όρους των ιδιοτήτων και των χειριστών. Τα χαρακτηριστικά ενός αντικειμένου δεν είναι απευθείας προσβάσιμα στο υπόλοιπο πρόγραμμα. Παρόλο που ένα αντικείμενο μιας κλάσης μπορεί να κληρονομήσει ιδιότητες και χειριστές από τις υπερκλάσεις, οι τιμές τους στο αντικείμενο δεν μπορούν να αλλάξουν χωρίς να σταλεί ένα μήνυμα στο ίδιο το αντικείμενο. Δηλαδή η COOL υποστηρίζει την ιδιότητα του εγκλεισμού, απαιτώντας την αποστολή μηνυμάτων (messages) για το χειρισμό στιγμιότυπων των κλάσεων, ορισμένων από το χρήστη αφού ,ένα στιγμιότυπο δε μπορεί να αποκριθεί σε ένα μήνυμα για το οποίο δεν έχει καθορισμένη μέθοδο. Η κλάση-ρίζα ή αλλιώς ρίζα της CLIPS είναι ένα προκαθορισμένο σύστημα κλάσεων που καλείται «Αντικείμενο» (OBJECT). Το προκαθορισμένο σύστημα κλάσεων “USER” είναι μια υποκλάση του “OBJECT” (Joseph C. Giarratano **2002,** Ι. Βλαχάβας και συνεργάτες **2006**).

Έχοντας δει τις έννοιες των πέντε βασικών στοιχείων της γλώσσας COOL θα δειχτεί ένα παράδειγμα για περισσότερη κατανόηση. Ορίζεται μια κλάση “HEROES” όπου γίνεται αναφορά σε ήρωες κόμικς. Η

Εικόνα 2.3.8.1.1 δείχνει πως οι “HEROES” παίρνουν την κληρονομικότητα τους από την κλάση-ρίζα “OBJECT”. Η κλάση “HEROES” ορίζεται ως υποκλάση της “USER”. Τα κουτιά ή οι κόμβοι χρησιμοποιούνται για την αναπαράσταση κλάσεων ενώ τα συνδετικά βέλη ονομάζονται **συνδέσεις** (links). Συνήθως βέβαια προτιμούνται απλές γραμμές για λόγους ευκολίας στην σχεδίαση. Εφόσον η CLIPS υποστηρίζει μόνο “is-a” συνδέσεις, η σχέση “is-a” δεν θα γράφεται δίπλα από κάθε σύνδεση από δω και στο εξής. Παρατηρείται ότι η ουρά του βέλους ξεκινά από την υποκλάση ενώ το κεφάλι του δείχνει προς την κλάση γονέα της. Η “is-a” σύνδεση υποδεικνύει την κληρονομικότητα των slots από μια κλάση προς την υποκλάση. Μια κλάση μπορεί να έχει από καθόλου έως πολλές υποκλάσεις. Όλες οι κλάσεις εκτός από την “OBJECT” πρέπει να έχουν μια υπερκλάση. Εφόσον η “HEROES” κληρονομεί όλα τα slots της “USER” και η “USER” κληρονομεί όλες τις ιδιότητες της “OBJECT”, έτσι η “HEROES” κληρονομεί και όλες τις ιδιότητες της “OBJECT”. Η ίδια αρχή ισχύει και για τους χείριστες μηνυμάτων. Δηλαδή η “HEROES” κληρονομεί τους χείριστες μηνυμάτων τόσο της “USER” όσο και της “OBJECT”.



Εικόνα 2.3.8.3.1 Παράδειγμα κληρονομικότητας (Joseph C. Giarratano **2002**)

Η κληρονομικότητα των ιδιοτήτων και των χειριστών είναι σημαντική αφού σημαίνει ότι δεν χρειάζεται να ανακαθοριστούν οι ιδιότητες και η συμπεριφορά των νέων κλάσεων των αντικειμένων που ορίζονται. Κατά συνέπεια κάθε νέα κλάση κληρονομεί τις ιδιότητες και την συμπεριφορά της αμέσως ανώτερης της από άποψη επιπέδου κλάσης. Για να οριστούν κλάσεις στην CLIPS χρησιμοποιείται η εντολή **(defclass).** Για να οριστεί η κλάση “HEROES” :

(defclass HEROES(is-a USER))

Η βασική δομή της εντολής “defclass” για να ορίζονται μόνο κλάσεις και όχι ιδιότητες, είναι:

(defclass <class> (is-a <direct-superclasses>))

Το **<direct-superclasses>** καλείται λίστα υπερκλάσεων επειδή καθορίζει το πώς μια κλάση συνδέεται με τις απευθείας υπερκλάσεις της. Δεν είναι απαραίτητο να μπαίνει μόνο ένα όνομα υπερκλάσης. Για παράδειγμα για τους κακούς που έχουν να αντιμετωπίσουν οι ήρωες:

(defclass VILLAINS (is-a PEOPLE COMIC USER OBJECT))

που σημαίνει ότι η κλάση “VILLAINS” έχει τέσσερις υπερκλάσεις με σειρά κληρονομικότητας ιδιοτήτων και χειριστών από το χαμηλότερο επίπεδο στο υψηλότερο (PEOPLE, COMIC, USER, OBJECT). Η λίστα υπερκλάσεων υποδηλώνει προτεραιότητα στην κληρονόμηση ιδιοτήτων και μεθόδων:

```
(defclass A
        (is-a B C D)
        ...
)
```

Προτεραιότητα στον ορισμό ιδιοτήτων και μεθόδων έχει η κλάση A ενώ στη συνέχεια η προτεραιότητα στην κληρονόμηση ορίζεται με τη σειρά (B C D). Πιο αναλυτικά η “defclass” αποτελείται από πέντε στοιχεία (Ι. Βλαχάβας και συνεργάτες **2006**).

- Όνομα της κλάσης
- Λίστα υπερκλάσεων από την οποία η νέα κλάση κληρονομεί ιδιότητες και μεθόδους
- Προσδιοριστής, ο οποίος καθορίζει εάν θα δημιουργούνται άμεσα αντικείμενα της νέας κλάσης
- Προσδιοριστής, ο οποίος καθορίζει εάν τα αντικείμενα αυτής της κλάσης μπορούν να ταιριάζουν με πρότυπα αντικειμένων στο αριστερό μέρος των κανόνων
- Λίστα ιδιοτήτων που ορίζονται στη νέα κλάση

Δηλαδή η πιο ειδική μορφή μιας κλάσης ορίζεται έτσι:

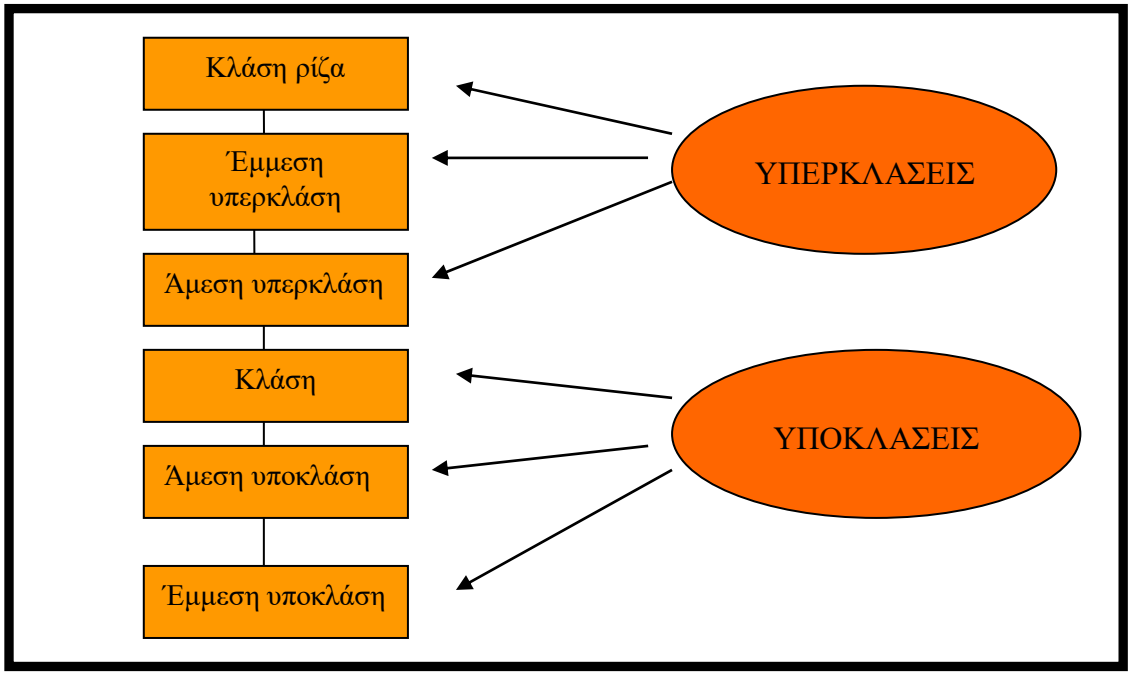
```
(defclass <όνομα> [<σχόλιο>] (is-a <όνομα υπερκλάσης>+)
        [<role>] [<pattern-match-role>] <ιδιότητες>*
        <handler-documentation>*
    )
```

Παράδειγμα

```
(defclass vehicle
            (is-a USER)
            (slot fuel-type (type SYMBOL))
            (slot tank-capacity (type INTEGER))
```

(slot fuel-loaded (type INTEGER)))

Επίσης αξιοσημείωτο είναι το γεγονός οι υποκλάσεις χωρίζονται σε δυο κατηγόριες, τις άμεσες και τις έμμεσες. Μια άμεση υποκλάση συνδέεται με μια μοναδική σύνδεση απευθείας σε μια υπερκλάση ενώ μια έμμεση έχει περισσότερες από μια συνδέσεις (Joseph C. Giarratano **2002**) (Εικόνα 2.3.8.3.2)



Εικόνα 2.3.8.3.2 Σχέσεις μεταξύ των κλάσεων (Joseph C. Giarratano **2002**)

Η κλάση ρίζα είναι η μοναδική που δεν έχει υπερκλάση. Είναι υπερκλάση όλων των άλλων, συμπεριλαμβανομένων και των ορισμένων από το χρήστη κλάσεων. Χρησιμοποιώντας την νέα αυτή ορολογία μας επιτρέπεται να πούμε ότι η βασική αρχή της κληρονομικότητας είναι :

Μια κλάση μπορεί να κληρονομήσει στοιχεία από όλες τις υπερκλάσεις της

Αυτή η αρχή δηλώνει ότι οι ιδιότητες και οι χειριστές μηνυμάτων μπορούν να κληρονομηθούν και να σώσουν κάποιον από τον κόπο του επανορισμού τους για τις νέες υποκλάσεις. Επιπρόσθετα, οι ιδιότητες μπορούν εύκολα να ταξινομηθούν σε νέες υποκλάσεις ως τροποποιήσεις και ως συνθετικά των ιδιοτήτων των υπερκλάσεων. Ως αποτέλεσμα σώζεται πολύτιμος χρόνος ανάπτυξης των προγραμμάτων ενώ το κόστος μειώνεται σημαντικά (Joseph C. Giarratano **2002**).

Μια κλάση μπορεί να είναι **αφηρημένη** (abstruct) ή **συγκεκριμένη** (concrete). Η αφηρημένη κλάση χρησιμοποιείται μόνο για λόγους κληρονόμησης ιδιοτήτων και μεθόδων σε υποκλάσεις της, και δεν έχει άμεσα στιγμιότυπα ενώ η συγκεκριμένη έχει άμεσα στιγμιότυπα και εάν δεν υπάρχει ο συγκεκριμένος προσδιοριστής στον ορισμό μιας κλάσης, αυτός καθορίζεται μέσω της κληρονομικότητας.

Οι ιδιότητες αποθηκεύουν τιμές που σχετίζονται με τα στιγμιότυπα-αντικείμενα κάθε κλάσης. Το όνομα της ιδιότητας μπορεί να είναι οποιοδήποτε σύμβολο, εκτός από “is-a” και “name” που χρησιμοποιούνται στις συνθήκες των κανόνων. Υπάρχουν δύο είδη ιδιοτήτων: οι απλής τιμής (single-slot) και οι πολλαπλών τιμών (multislot). Κάθε αντικείμενο έχει ένα αντίγραφο από τις ιδιότητες της άμεσης κλάσης του, καθώς και των ιδιοτήτων που κληρονομεί η κλάση. Οι ιδιότητες κληρονομούνται από τις κλάσεις με τη σειρά που ορίζεται από τη λίστα προτεραιότητας κλάσεων. Εάν μια ιδιότητα κληρονομείται από δύο διαφορετικές κλάσεις, τότε χρησιμοποιείται ο ορισμός από τη πιο συγκεκριμένη κλάση, δηλαδή αυτή που βρίσκεται πιο αριστερά στη λίστα προτεραιότητας. Υπάρχουν και ιδιότητες που δεν κληρονομούνται λόγω της δήλωσης “no-inherit” (Ι. Βλαχάβας και συνεργάτες **2006**).

Τα αντικείμενα ή στιγμιότυπα όπως και τα γεγονότα δημιουργούνται από τον χρήστη, μπορούν να διαγραφτούν από την εντολή (reset) αλλά και να φορτωθούν από ένα αρχείο. Για να δημιουργηθεί ένα αντικείμενο χρησιμοποιείται η συνάρτηση (**make-instance).**

Παράδειγμα

(make-instance
[<όνομα στιγμιότυπου>]of <όνομα κλάσης>
(<όνομα ιδιότητας> <expression>*)*

Το όνομα του αντικειμένου περικλείεται από αγκύλες οι οποίες ωστόσο δεν αποτελούν μέρος του ονόματος αλλά θεωρούνται ένδειξη ότι πρόκειται για όνομα αντικειμένου. Η συνάρτηση (make-instance) επιστρέφει το όνομα του νέου αντικειμένου ή το σύμβολο “FALSE” εάν υπάρξει αποτυχία. Το όνομα του στιγμιότυπου μπορεί να είναι είτε τύπου “SYMBOL” είτε τύπου “INSTANCE-NAME”**.** Αν το όνομα δεν καθορίζεται στην (make-instance) τότε η CLIPS δημιουργεί ένα όνομα καλώντας τη συνάρτηση (**gensym*).** Αν το αντικείμενο υπάρχει ήδη τότε η (make-instance) το διαγράφει πρώτα και το ξαναδημιουργεί με τις νέες τιμές. Με τη βοήθεια της έκφρασης (**definstances**) ορίζονται αντικείμενα μέσα σε ένα αρχείο και δημιουργούνται κάθε φορά που εκτελείται η εντολή (reset) (Ι. Βλαχάβας και συνεργάτες **2006**)**:**

(definstances <definstances-name> [<comment>]
([<instance-name-expression>] of <class-name-expression>
(<slot-name-expression> <expression>*)*)*)

Εδώ πρέπει οι κλάσεις να έχουν ήδη οριστεί και εάν αποτύχει η δημιουργία κάποιου αντικειμένου, τότε τα υπόλοιπα αντικείμενα αγνοούνται. Η (definstances) χρησιμοποιεί εσωτερικά τη συνάρτηση (make-instance).

(definstances car
(fiat_brava of car
(fuel-type benzine)
(tank-capacity 45)
(fuel-loaded 30)
(consumption-rate 10)
(reset-counter 250)))

Τα στιγμιότυπα μπορούν να χειριστούν με αποστολή μηνυμάτων σε αυτά με τη χρήση της συνάρτησης (**send**). Οι παράμετροι που χρησιμοποιούνται μέσω αυτής της συνάρτησης είναι: αντικείμενο-παραλήπτης μηνύματος, όνομα μηνύματος, παράμετροι μηνύματος.

(send <object-expression>
<message-name-expression> <expression>*)

Η συνάρτηση (send) επιστρέφει ως τιμή το αποτέλεσμα του μηνύματος:

(send [fiat_brava] put-fuel-type petroleum)

Αποτέλεσμα: petroleum

2.3.9 Πληροφορίες

Η Clips έχει και "online" βοήθεια διαθέσιμη σε περίπτωση που κάποιος δεν μπορέσει να κατανοήσει κάτι. Για να αποκτήσει κάποιος πρόσβαση αρκεί να εισάγει την εντολή **(help)** και να πατήσει "enter". Σε σύντομο χρονικό διάστημα θα εμφανιστεί ένα μενού επιλογών.

Απόγονοι της CLIPS γλώσσας είναι το JESS (Java Expert System Shell) και η ασαφής CLIPS. Το Jess είναι μια μηχανή εξαγωγής συμπερασμάτων για τη πλατφόρμα της Java και μια ανώτερη εκδοχή της CLIPS. Αναπτύχθηκε από τον Ernest Friedman-Hill στο τέλος της χρονιάς του 1995. Παρέχει προγραμματισμό βασισμένο σε κανόνες κατάλληλο για την αυτοματοποίηση ενός έμπειρου συστήματος και συχνά αναφέρεται και σαν κέλυφος έμπειρου συστήματος. Η μηχανή εξαγωγής συμπερασμάτων του Jess χρησιμοποιεί τον αλγόριθμο Rete. Το Jess χρησιμοποιείται σε εφαρμογές οι οποίες χρησιμοποιούν γνώση με την μορφή κανόνων δηλώσεων ώστε να βγάλουν συμπεράσματα και να πραγματοποιήσουν παρεμβολές. Οι κανόνες μπορούν να τροποποιήσουν το σετ των γεγονότων ή μπορούν να εκτελέσουν οποιοδήποτε κώδικα Java (http://herzberg.ca.sandia.gov/, http://en.wikipedia.org/wiki/Jess_programming_language).

Η ασαφής CLIPS είναι μια ασαφής λογική προέκταση της CLIPS δημιουργημένη από τη Nasa. Αναπτύχθηκε από το Integrated Reasoning Group του Ινστιτούτου Πληροφορίας Τεχνολογίας του Εθνικού Συμβουλίου Ερευνάς του Καναδά και διανέμεται ευρέως για πολλά χρόνια. Ενσωματώνει τη CLIPS παρέχοντας μια δυνατότητα ασαφής λογικής η οποία είναι πλήρως ενσωματωμένη στα γεγονότα της CLIPS και στην μηχανή εξαγωγής συμπερασμάτων. Αντιμετωπίζει ακριβή, ασαφή ή και αβέβαιη λογική, επιτρέποντας ασαφής και λογικούς όρους να συνδυάζονται ελευθέρα στους κανόνες και στα γεγονότα ενός έμπειρου συστήματος που χρησιμοποιεί την ασάφεια και την αβεβαιότητα. Ως συνέπεια προμηθεύει ένα χρήσιμο περιβάλλον για την ανάπτυξη των ασαφών εφαρμογών ωστόσο απαιτεί σημαντική προσπάθεια για την ανανέωση και τη διατήρηση της αφού νέες εκδοχές της CLIPS κάνουν την εμφάνιση τους (http://en.wikipedia.org/wiki/FuzzyCLIPS).

Η τοποθεσία της γλώσσας CLIPS στο Διαδίκτυο είναι http://www.ghg.net/clips/CLIPS.html.

Κεφαλαίο 3: Περιγραφή του προβλήματος

3.1 Εισαγωγή

Στο πρώτο κεφάλαιο αναλύθηκε η έννοια του έμπειρου συστήματος και η χρησιμότητα του στον τομέα της υγείας ενώ στο δεύτερο κεφάλαιο έγινε επαρκής περιγραφή της γλώσσας δημιουργίας έμπειρων συστημάτων CLIPS. Στη συνέχεια με τη βοήθεια της θα δημιουργηθεί ένα έμπειρο σύστημα που θα βοηθάει στη διάγνωση αλλεργιών με βάση τα συμπτώματα που παρουσιάζει ο ασθενής. Ως συνέπεια, με ένα πολύ εύκολο και πρακτικό τρόπο θα αναγνωρίζεται εάν κάποιο άτομο έχει αλλεργία. Τα έμπειρα συστήματα είναι ουσιαστικά, εργαλεία που βοηθούν στη διάγνωση όταν ο γιατρός είναι άπειρος ή αβέβαιος καθώς και στη κλινική εκπαίδευση του.

Πιο αναλυτικά όμως τι είναι η αλλεργία; Η αλλεργία είναι μία λέξη που χρησιμοποιείται για τη περιγραφή μίας πληθώρας συμπτωμάτων, από τη κρίση φτερνίσματος και δακρύρροιας έως το κόκκινο εξάνθημα που προκαλεί κνησμό, τα οποία παρουσιάζονται μετά το χάιδεμα ενός ζώου (συνήθως κατοικίδιου) ή το οίδημα των χειλιών και το μούδιασμα του στόματος που εμφανίζονται μετά τη κατανάλωση μιας συγκεκριμένης ουσίας. "Αλλεργία" είναι μια ακατάλληλη και επιζήμια αντίδραση του αμυντικού μηχανισμού του σώματος σε ουσίες που είναι φυσιολογικά αβλαβείς (http://health.in.gr/allergies/default.asp).

Μία αλλεργία μπορεί να εκδηλωθεί με διαφορετικούς τρόπους, σε διάφορα όργανα του σώματος. Μπορεί επίσης, να είναι περισσότερο ή λιγότερο σοβαρή. Είναι λοιπόν σημαντικό να είναι κάποιος ενημερωμένος ώστε να έχει την ικανότητα να αναγνωρίζει τα συμπτώματα των αλλεργιών στην αρχική τους φάση, με σκοπό να μπορεί να προσφέρει την ανάλογη βοήθεια και να αποφύγει τυχόν τραγικές καταστάσεις.

3.1.1 Είδη αλλεργιών

Υπάρχουν διάφορων ειδών αλλεργίες όπου οι αιτίες εμφάνισης τους ποικίλλουν. Θεωρήθηκε εύστοχο να καταταχτούν οι αλλεργίες σε τροφικές, σε ρινικές και σε δερματικές. Οι κρίσεις φτερνίσματος, καταρροής και αποφραγμένης μύτης, σε συνδυασμό με μάτια με κνησμό και δακρύρροια ονομάστηκαν για πρώτη φορά «αλλεργική ρινίτιδα» στα μέσα του 19ου αιώνα. Το κύριο αίτιο είναι η αλλεργία στη γύρη. Τα συμπτώματα ωστόσο μπορεί να εκδηλώνονται καθ' όλη τη διάρκεια του χρόνου και τα βασικά αλλεργιογόνα που συσχετίζονται είναι από τα ακάρεα της οικιακής σκόνης και τα κατοικίδια. Η αλλεργική ρινίτιδα είναι η πιο συχνή αλλεργική πάθηση αφού προσβάλλει σήμερα το 10% του πληθυσμού. Δεν επηρεάζεται μόνο η περιοχή της μύτης και ένα από τα πιο ενοχλητικά συμπτώματα είναι τα μάτια που έχουν κνησμό και δακρύζουν ασταμάτητα. Επιπλέον, πέρα από αυτά τα συμπτώματα μπορεί να εκδηλωθεί και δύσπνοια. Συνήθως για να αντιμετωπιστεί χρησιμοποιούνται αντιισταμινικά, αποσυμφορητικά και κορτικοστεροειδή

φάρμακα που συμβάλλουν στην ανακούφιση του πάσχον (http://health.in.gr/allergies/default.asp).

Επιπλέον οι αλλεργικές αντιδράσεις σε τροφές είναι εξαιρετικά συχνές, αν και μόνο ένα πολύ μικρό ποσοστό των συμπτωμάτων που εκδηλώνονται μετά την κατανάλωση συγκεκριμένων τροφών είναι πραγματικές αλλεργίες- συμμετέχει, δηλαδή, σε αυτές το ανοσοποιητικό σύστημα του οργανισμού. Οι τροφικές αλλεργίες είναι εξαιρετικά σπάνιες. Οι αντιδράσεις στις τροφές μπορεί να οφείλονται στην έλλειψη ενός συγκεκριμένου πεπτικού ενζύμου. Η τροφική αλλεργία- κυρίως στο γάλα, τα αυγά, τα ψάρια και τους ξηρούς καρπούς- μπορεί να προκαλέσει σοβαρά συμπτώματα, όπως οίδημα των χειλιών και της γλώσσας, συριγμό και χαμηλή αρτηριακή πίεση. Ο καλύτερος τρόπος αντιμετώπισης είναι η αποφυγή της συγκεκριμένης τροφής ενώ οι σοβαρότερες αντιδράσεις πρέπει να αντιμετωπίζονται με χορήγηση αδρεναλίνης. Μερικές φορές, τα συμπτώματα της τροφικής αλλεργίας μπορεί να εμφανιστούν πιο συγκαλυμμένα και η διάγνωσή τους μπορεί να είναι πιο δύσκολη, απαιτώντας την εξειδικευμένη βοήθεια ενός διαιτολόγου (http://health.in.gr/allergies/default.asp).

Επίσης διακρίνονται αλλεργίες που σχετίζονται με δερματικές διαταραχές που προκαλούν κνησμό εξαιτίας της επαφής του δέρματος με ορισμένες ουσίες. Το δέρμα ερεθίζεται και ως αποτέλεσμα προκαλείται κνησμός ενώ δεν αποκλείεται η πιθανότητα εμφάνισης φλυκταινών. Υπάρχουν πολλές ουσίες που μπορεί να «πυροδοτήσουν» μία κρίση, περιλαμβανομένων ορισμένων τροφών, φυτών και καλλυντικών. Τα συνήθη αλλεργιογόνα εδώ είναι το νικέλιο (αγκράφες ρούχων, σκελετοί γυαλιών, κοσμήματα, νομίσματα, μαγειρικά σκεύη), το χρωμικό άλας (δέρμα, λευκαντικά, σπίρτα, τσιμέντο), η Φορμαλδεΰδη (συντηρητικά, καλλυντικά, βερνίκια νυχιών, δημοσιογραφικό χαρτί, τσιγάρα, μαλακτικά ρούχων, ρούχα που αντέχουν στο ζάρωμα), η Αιθυλενοδιαμίνη (συντηρητικά σε κρέμες, μπογιές) το Μερκαπτοβενζοθειαζόλιο (προϊόντα από φυσικό λάστιχο, ιδίως μπότες, γάντια και καθετήρες), η Θειαβενταζόλη (προϊόντα από λάστιχο, μυκητοκτόνες ουσίες σε μπογιές και σαπούνια) και τα φυτά Ηράνθεμα, δηλητηριώδης κισσός, ντάλιες και χρυσάνθεμα (http://health.in.gr/allergies/default.asp).

3.2 Περιγραφή του προβλήματος

Για το συγκεκριμένο πρόβλημα έγινε συλλογή και διαχωρισμός των πιο κοινών συμπτωμάτων που προκύπτουν όταν κάποιος άνθρωπος έχει αλλεργία (οποιασδήποτε κατηγόριας). Το μούδιασμα στόματος, το πρήξιμο των χειλιών, ο εμετός, η ξηροδερμία καθώς και το ξεφλούδισμα δέρματος αποτελούν παρενέργειες συνήθως τροφικών αλλεργιών. Επίσης αλλά συμπτώματα που μπορούν να προκύψουν από τσιμπήματα, σκόνη, γύρη, κατοικίδια ζώα, μυρωδιές ή υλικά είναι η φαγούρα στη μύτη, το μπούκωμα και η καταρροή μύτης, η δυσκολία στην αναπνοή, τα εξανθήματα, η αναφυλαξία, το έντονο φτέρνισμα, η δακρύρροια και ο κνησμός λαιμού και ματιών. Δυστυχώς πολλά τέτοια συμπτώματα όμως μπερδεύονται με άλλες παθήσεις όπως η ιλαρά και η γρίπη και οι αλλεργίες δεν αναγνωρίζονται αμέσως. Γι αυτό είναι πολύ σημαντικό να γίνεται διαχωρισμός συμπτωμάτων που

αποτελούν ενδείξεις αλλεργίας ή όχι γιατί εάν δεν είναι κάποιος προσεκτικός μπορεί να οδηγηθεί και σε απώλεια ανθρώπινης ζωής.

Θεωρήθηκε απαραίτητο να εισαχτεί η έννοια της ασθένειας της ιλαράς καθώς και της γρίπης διότι τα συμπτώματα τους όπως προαναφέρθηκε μοιάζουν με κάποια συμπτώματα αλλεργιών. Έτσι θα αποφευχθούν τυχόν συγχύσεις σε ασθενείς οι οποίοι ενδεχομένως να μην έχουν αλλεργία ή ακριβώς το ανάποδο, για να γίνεται σωστή χορήγηση φάρμακων. Γι' αυτόν ακριβώς τον λόγο δημιουργήθηκε ένας κανόνας που ζητά ως είσοδο τη θερμοκρασία του ασθενή και του δόθηκε προτεραιότητα από τους υπόλοιπους ώστε να γίνεται στην αρχή η διάγνωση για τυχόν γρίπη ή ιλαρά εφόσον ο ασθενής εμφανίζει σημάδια πυρετού.

Προκειμένου να υλοποιηθεί το συγκεκριμένο έμπειρο σύστημα διάγνωσης αλλεργιών συντάχθηκαν ερωτήσεις μέσω πρότυπων γεγονότων. Ωστόσο το έμπειρο σύστημα το οποίο θα δημιουργηθεί έχει βασιστεί πάνω σε μια πολύ απλή εφαρμογή διάγνωσης αλλεργιών (http://www.cis.ysu.edu/~john/824/examples.html). Μέσα από όλη αυτή την διαδικασία στόχος είναι η αναδιαμόρφωση του συστήματος, σχεδιάζοντας και υλοποιώντας ένα ανώτερο και πιο πολύπλοκο σε επίπεδο κώδικα και αποτελεσμάτων σύστημα. Τα πρότυπα γεγονότων βοηθούν στην εύκολη δομή του προγράμματος δίνοντας παράλληλα ένα ιδιαίτερο στυλ. Επιλέχθηκαν ως κύρια μορφή αναπαράστασης των σύνθετων γεγονότων που αποτελούν τα συμπτώματα και οι διάφορες κατηγόριες αλλεργιών. Στη πορεία διαπιστώθηκε ότι με τη χρήση αυτών μπορούν να πραγματοποιηθούν ερωτήσεις και έχοντας ορίσει προκαθορισμένες και ιατρικά επιβεβαιωμένες απαντήσεις, εξάγονται ξεκάθαρα συμπεράσματα εάν κάποιος έχει αλλεργία ή όχι. Οι απαντήσεις που μπορούν να δοθούν είναι «ναι»(yes) ή «όχι»(no) πάνω σε ερωτήσεις για τι πιθανά συμπτώματα έχει ο ασθενής. Κάθε φορά ο συνδυασμός των απαντήσεων που δίνεται αντιστοιχίζεται με μια λίστα πιθανών διαγνώσεων ανάλογα με τις απαντήσεις που παρέχονται, παίρνοντας έτσι την αντίστοιχη διάγνωση και θεραπεία. Να σημειωθεί ότι μόλις πραγματοποιηθεί μια διάγνωση, προτείνεται αμέσως θεραπεία και σε αυτό το σημείο σταματά η διαδικασία των ερωτήσεων.

Το πρόγραμμα αποτελείται από τέσσερα μέρη. Το πρώτο αφορά τα πρότυπα γεγονότων, το δεύτερο την ομάδα των κανόνων ερωτήσεων, το τρίτο την ομάδα των κανόνων που αφορούν τη διάγνωση και το τέταρτο την ομάδα των κανόνων που αφορούν τη προτεινόμενη θεραπεία.

3.2.1 Πρότυπα γεγονότων: πρώτο μέρος

Πιο αναλυτικά δημιουργήθηκαν είκοσι πρότυπα γεγονότων. Τα δεκαοχτώ από τα είκοσι αποτελούν το καθένα ξεχωριστά ένα σύμπτωμα αλλεργίας, γρίπης και ιλαράς και έχουν ως ιδιότητα τα αντίστοιχα συμπτώματα. Επίσης θεωρήθηκε εύστοχο να δημιουργηθούν δυο πρότυπα γεγονότων ένα για τη διάγνωση και ένα για τη θεραπεία που προτείνεται από την στιγμή που έχει διαπιστωθεί τι έχει ο ασθενής. Αξιοσημείωτο είναι το γεγονός ότι σχεδιάστηκε και ένα πρότυπο για τον πυρετό το οποίο παίρνει σαν ιδιότητα το επίπεδο του πυρετού που μπορεί να έχει ο ασθενής.

Κλείνοντας φτιάχτηκε και ένα πρότυπο για τη θερμοκρασία του ασθενή που σχετίζεται με το εάν έχει πυρετό. Παρακάτω δίνεται η λίστα όλων των πρότυπων που χρησιμοποιούνται στο έμπειρο σύστημα διάγνωσης αλλεργιών:

```
(deftemplate Spots (slot spots))

(deftemplate Rash (slot rash))

(deftemplate SoreThroat  (slot sore_throat))

(deftemplate Innoculated  (slot innoculated))

(deftemplate DrySkin  (slot dry_skin))

(deftemplate ItsiNose  (slot itsi_nose))

(deftemplate Flake (slot flake))

(deftemplate RunningNose ( slot running_nose))

(deftemplate Congest  (slot congest))

(deftemplate IntenseSneez  (slot intense_sneez))

(deftemplate DifficultyInBreathing  (slot
difficulty_in_breathing))

(deftemplate NumbedMouth  (slot numbed_mouth))

(deftemplate BumpedLips  (slot bumped_lips))

(deftemplate ItsiEyes  (slot itsi_eyes))

(deftemplate IntenseTears (slot intense_tears))

(deftemplate Diagnosis   (slot diagnosis))

(deftemplate Treatment   (slot treatment))

(deftemplate Fever   (slot level))

(deftemplate Temperature (slot temperature))

(deftemplate Vomit (slot vomit))
```

3.2.2 Κανόνες ερωτήσεων: δεύτερο μέρος

Αφού δημιουργήθηκαν τα πρότυπα γεγονότων, έπειτα σχεδιάζεται μία ομάδα κανόνων defrule Get (ΌνομαΣυμπτώματος) οι οποίοι ονομάζονται κανόνες ερωτήσεων. Μέσω αυτών των κανόνων πραγματοποιούνται ερωτήσεις για τα πρότυπα γεγονότων που έχουν οριστεί παραπάνω. Με άλλα λόγια ερωτάται εάν ο ασθενής έχει κάποιο σύμπτωμα ή όχι. Οι απαντήσεις που μπορούν να δοθούν είναι είτε καταφατικές «ναι» είτε αρνητικές «όχι». Ανάλογα με τον συνδυασμό των απαντήσεων ενεργοποιείται ο αντίστοιχος κανόνας διάγνωσης και θεραπείας (3.2.4). Η δομή των κανόνων αυτών είναι ίδια σε όλους. Δεν χρησιμοποιείται τίποτα σαν εντολή του αριστερού μέρους αλλά αντιθέτως εισάγεται η συνάρτηση “printout” στο δεξί μέρος του κανόνα. Ως αποτέλεσμα τυπώνεται η ερώτηση στην οθόνη και το σύστημα αναμένει απάντηση. Η δοθούμενη απάντηση δεσμεύεται στη μεταβλητή “?response” και διαβάζεται μέσω της συνάρτησης “read”. Για να κλείσει και να εκτελεστεί επιτυχώς ο κανόνας, πραγματοποιείται η εισαγωγή της απάντησης που δόθηκε υπό την μορφή της μεταβλητής “?response” στην αντίστοιχη ιδιότητα του αντίστοιχου πρότυπου. Να τονιστεί εδώ ότι δημιουργήθηκε ένας κανόνας που ζητά τη θερμοκρασία του ασθενή ώστε να διαπιστωθεί εάν έχει πυρετό που με το συνδυασμό συγκεκριμένων συμπτωμάτων μπορεί να πάσχει από ιλαρά ή ακόμη και από γρίπη και όχι από αλλεργία. Για την ικανοποίηση του κανόνα διάγνωσης της ιλαράς δημιουργήθηκε ένας επιπλέον κανόνας που ερωτά εάν ο ασθενής έχει εμβολιαστεί για την ιλαρά. Πιο συγκεκριμένα οι κανόνες που συντάχθηκαν με σκοπό να ερωτά το σύστημα εάν ο ασθενής παρουσιάζει κάποια συμπτώματα (Ιλαράς, αλλεργίας, γρίπης) ή όχι είναι οι εξής:

```
(defrule GetTemperature
   =>
   (printout t "Please give the patient's temperature in
order to see if he has fever or not:  ")
   (bind ?response (read))
   (assert (Temperature (temperature ?response))))


(defrule GetSpots
   =>
   (printout t "Does the patient have spots on his face
or on his body? (yes or no):  ")
   (bind ?response (read))
   (assert (Spots (spots ?response))))


(defrule GetInnoculated
   (Fever (level high))
   (Spots (spots yes))
   =>
```

```
   (printout t "Has the patient been innoculated for
measles or he was afraid the niddles? (yes or no):  ")
   (bind ?response (read))
   (assert (Innoculated (innoculated ?response))))


(defrule GetIntenseTears
=>
(printout t "Does the patient have too much tears-like he
is crying a lot for no reason? (yes or no): ")
(bind ?response (read))
(assert(IntenseTears(intense_tears ?response))))


(defrule GetRash
   =>
   (printout t "Does the patient have or feel a
rash(hour) generally? (yes or no):  ")
   (bind ?response (read))
   (assert (Rash (rash ?response))))


(defrule GetItsiEyes
=>
(printout t "Does the patient have the desire to scratch
his eyes because he feels them itsi(&scratsy) or hottie?
(yes or no):  ")
(bind ?response (read))
(assert(ItsiEyes (itsi_eyes ?response))))


(defrule GetSoreThroat
   =>
   (printout t "Does the patient have a sore throat? (yes
or no):  ")
   (bind ?response (read))
   (assert (SoreThroat (sore_throat ?response))))


(defrule GetItsiNose
=>
(printout t "Does the patient have itsi nose? (yes or no)
:  ")
(bind ?response (read))
(assert (ItsiNose(itsi_nose ?response))))


(defrule GetCongest
=>
(printout t " Does the patient has congest on his nose?
(yes or no) :  ")
```

```
(bind ?response (read))
(assert (Congest(congest ?response))))


(defrule GetDrySkin
=>
(printout t "Does the patient have dry skin? (yes or no):
")
(bind ?response (read))
(assert(DrySkin(dry_skin ?response))))


(defrule GetFlake
=>
(printout t "Does the patient have (corn)flake skin? (yes
or no) :  ")
(bind ?response (read))
(assert(Flake(flake ?response))))


(defrule GetNumbedMouth
=>
(printout t "Does the patient feel a numbing around his
mouth? (yes or no):  ")
(bind ?response (read))
(assert(NumbedMouth(numbed_mouth ?response))))


(defrule GetIntenseSneez
=>
(printout t "Does the patient have an intense sneez? (yes
or no):  ")
(bind ?response (read))
(assert(IntenseSneez(intense_sneez ?response))))


(defrule GetDifficultyInBreathing
=>
(printout t "Does the patient face difficulties in
breathing? (yes or no) :  ")
(bind ?response (read))
(assert(DifficultyInBreathing(difficulty_in_breathing
?response))))


(defrule GetRunningNose
=>
  (printout t "Does the patient have a running nose? (yes
or no) :  ")
  (bind ?response (read))
  (assert(RunningNose(running_nose ?response))))
```

```
(defrule GetBumpedLips
=>
(printout t "Does the patient have bumped lips? (yes or
no):  ")
(bind ?response (read))
(assert(BumpedLips (bumped_lips ?response))))


(defrule GetVomit
=>
(printout t "Does the patient have the feeling of
vomiting or he has already vomit? (yes or no):  ")
(bind ?response (read))
(assert(Vomit (vomit ?response))))
```

3.2.3 Κανόνες διάγνωσης αλλεργίας: τρίτο μέρος

Μετά τη δημιουργία των κανόνων ερωτήσεων για τα συμπτώματα που μπορεί να έχει ο ασθενής, δημιουργείται η επόμενη ομάδα κανόνων που ονομάζονται κανόνες διάγνωσης. Μέσω αυτών διακρίνεται εάν ο ασθενής έχει δερματική, τροφική αλλεργία, αλλεργική ρινίτιδα, ιλαρά ή γρίπη με βάση τα συμπτώματα που παρουσιάζονται. Η δομή των κανόνων διάγνωσης αποτελείται στο αριστερό τους μέρος από συνθήκες. Αυτές είναι τα συμπτώματα που εμφανίζονται ή όχι στον ασθενή. Όταν αυτές ικανοποιούνται (δηλαδή ο ασθενής εμφανίζει κάποια συμπτώματα που ικανοποιούν το αριστερό μέρος ενός κανόνα διάγνωσης) τότε πραγματοποιείται μια εισαγωγή στην ιδιότητα του πρότυπου της διάγνωσης με το όνομα της ασθένειας (αλλεργία, τροφική αλλεργία, γρίπη, ιλαρά, αλλεργική ρινίτιδα) που παρουσιάζει να έχει ο ασθενής με βάση τα συμπτώματα που έχει ή ότι δεν έχει απαντώντας στις ερωτήσεις του έμπειρου συστήματος. Έπειτα εκτυπώνεται στην οθόνη με την βοήθεια της συνάρτησης “printout” η διάγνωση που έχει οριστεί και πραγματοποιείται έξοδος από τον κύκλο των ερωτήσεων αφού έχει διαπιστωθεί από τι πάσχει ο ασθενής (και έχει προταθεί η αντίστοιχη θεραπεία. Γι’ αυτό όμως θα μιλήσουμε στο τέταρτο μέρος). Η έξοδος από τον κύκλο των ερωτήσεων οφείλεται στην ύπαρξη της εντολής “set-break”. Χρησιμοποιείται η εντολή “set-break” με το όνομα του επόμενου κανόνα defrule Get (ΌνομαΣυμπτώματος) που βρίσκεται στην σειρά.. Δηλαδή για να γίνει πιο κατανοητό από την στιγμή που ενεργοποιείται ο κανόνας διάγνωσης και εκτυπώνεται στην οθόνη η διάγνωση και η αντίστοιχη θεραπεία, τότε πραγματοποιείται έξοδος από τις υπόλοιπες ερωτήσεις και εκεί είναι που κάνει την δουλεία της η “set-break”. «Σπάει» την σειρά των ερωτήσεων που γίνονται προς τον χρήστη τη στιγμή που επρόκειτο να γίνει η επόμενη ερώτηση χρησιμοποιώντας το όνομα του αντιστοίχου (επόμενου) κανόνα Get (ΌνομαΣυμπτώματος).

Αρχικά στη ομάδα των κανόνων διάγνωσης υπάρχουν τέσσερις κανόνες πυρετού. Για πιο λεπτομερή εξήγηση, έχουν δημιουργηθούν τέσσερις κανόνες μέσω των οποίων διαπιστώνεται εάν ο ασθενής έχει πυρετό. Ο πρώτος κανόνας υποδεικνύει ότι εάν η θερμοκρασία που θα δοθεί είναι ίση ή μεγαλύτερη από 38,5 °C τότε ο ασθενής έχει υψηλό πυρετό και το εξής μήνυμα θα εμφανιστεί στην οθόνη:

```
"Danger!!!! High fever diagnosed"
```

Αντίστοιχα για τους άλλους τρεις θα εμφανιστούν τα εξής μηνύματα:

```
"Mild fever diagnosed! Nothing to worry about.."
```
(37 °C <= θερμοκρασία <38.5 °C)

```
"The patient has no signs of fever! Everything is
superfine.."
```
(32 °C <= θερμοκρασία <37 °C)

```
"The patient has no signs of life! Is this a joke??
Although I am an expert system, you shouldn't make fun of
me! I have feelings too!!"
```
(θερμοκρασία <32 °C)

Στο αριστερό μέρος του κανόνα, ο κανόνας δέχεται την τιμή της θερμοκρασίας που θα δοθεί από τον χρήστη και εάν αυτή είναι μεγαλύτερη ή ίση από 38,5 °C τότε θα εισαχθεί στο πρότυπο του πυρετού στην ιδιότητα "level", η λέξη "high". Εδώ γίνεται χρήση της συνάρτηση "test" η οποία λειτουργεί σαν το "if"… "then". Η "test" καθορίζει εάν η θερμοκρασία είναι μεγαλύτερη ή ίση από 38,5 °C και πραγματοποιεί αντίστοιχα εισαγωγή του "high" στο πρότυπο του πυρετού. Ομοίως λειτουργούν και οι άλλοι τρεις κανόνες με την μόνη διάφορα ότι ο δεύτερος δίνει διάγνωση εάν ο ασθενής έχει μέσης έντασης πυρετό (κυμαίνεται από 38,5 °C έως 37 °C), ο τρίτος εάν ο ασθενής έχει φυσιολογική θερμοκρασία και ο τέταρτος αστειεύεται λέγοντας ότι άνθρωπος ζωντανός δεν νοείται να έχει τέτοια θερμοκρασία. Να σημειωθεί ότι και οι τέσσερις κανόνες κάνουν ορθή χρήση της συνάρτησης "test".

```
(defrule Fever1
   (Temperature (temperature ?t))
   (test (>=  ?t 38.5))
=>
   (assert (Fever (level high)))
   (printout t "Danger!!!! High fever diagnosed" crlf))


(defrule Fever2
   (Temperature (temperature ?t))
   (test (and (<  ?t  38.5) (>=  ?t  37)))
=>
   (assert (Fever (level mild)))
```

```
   (printout t "Mild fever diagnosed! Nothing to worry
about" crlf))


(defrule Fever3
   (Temperature (temperature ?t))
   (test (and (<  ?t  37) (>=  ?t 32)))
=>
   (assert(Fever(level normal)))
   (printout t " The patient has no signs of fever!
Everything is fine.." crlf))


(defrule Fever4
   (Temperature (temperature ?t))
   (test(<  ?t  32))
=>
   (assert(Fever(level paranormal)))
   (printout t "“The patient has no signs of life! Is
this a joke?? Although I am an expert system, you
shouldn’t make fun of me! I have feelings too!!” crlf))
```

Επιπλέον δημιουργήθηκε ο κανόνας “Measles” προκείμενου να γίνεται διαχωρισμός του ποτέ ένας ασθενής έχει αλλεργία και ποτέ ιλαρά. Όταν ο ασθενής δεν έχει κάνει εμβόλιο ιλαράς (innoculated no), εμφανίζει εξανθήματα και παρουσιάζει υψηλό πυρετό τότε θεωρείται ότι έχει ιλαρά. Είναι πολύ εύκολο αυτά τα συμπτώματα να ταυτιστούν με συμπτώματα αλλεργιών ή και ανάποδα και ως αποτέλεσμα να χαθεί κρίσιμος χρόνος αντιμετώπισης της αλλεργίας ή της ιλαράς ή να γίνει λάθος ιατρική χορήγηση. Σκόπιμα δίνεται προτεραιότητα στον κανόνα της διάγνωσης ιλαράς έτσι ώστε από τη στιγμή που κάποιος έχει πυρετό και εξανθήματα να ερωτάται αυτόματα εάν έχει εμβολιαστεί. Εάν ναι τότε πραγματοποιείται μετάβαση στον επόμενο κανόνα που αφορά την εμφάνιση δακρύρροιας στον ασθενή. Εάν όχι ο ασθενής πάσχει από ιλαρά εφόσον εμφανίζει συμπτώματα υψηλού πυρετού και εξανθήματα ενώ παράλληλα δεν έχει εμβολιαστεί γι’ αυτή την πάθηση.

```
(defrule Measles
   (declare (salience 100))
   (Spots (spots yes))
   (Innoculated (innoculated no))
   (Fever (level high))
 =>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles have been diagnosed for the
patient and this is not an allergy" crlf)
(set-break GetIntenseTears))
```

Στη συνέχεια δημιουργούνται κατά σειρά κανόνες διάγνωσης δερματικών αλλεργιών, αλλεργικής ρινίτιδας, γρίπης και τροφικής αλλεργίας

με διάφορους συνδυασμούς συμπτωμάτων οι οποίοι «προδίδουν» το είδος της αλλεργίας:

```
(defrule SkinAllergy1
   (Rash (rash yes))
   (Spots(spots yes))
   (SoreThroat(sore_throat yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy unfortunately has been
diagnosed from rash, sore throat and from the appearance
of spots" crlf)
   (set-break GetItsiNose))


(defrule SkinAllergy2
   (Spots(spots yes))
   (DrySkin(dry_skin yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy has been diagnosed from dry
skin and from the appearance of spots-I understand you
mate!" crlf)
   (set-break GetFlake))


(defrule SkinAllergy3
   (SoreThroat (sore_throat yes))
   (Rash(rash yes))
   (Flake(flake yes))
=>

   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy has been diagnosed from the
appearance of rash, flake and sore throating! Allergies
are pretty bad!" crlf)
    (set-break GetNumbedMouth))



(defrule SkinAllergy4
   (SoreThroat (sore_throat yes))
   (Rash(rash yes))
   (DrySkin (dry_skin yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy has been diagnosed from the
appearance rash, sore throat and dry skin! Allergies are
pretty bad!" crlf)
    (set-break GetFlake))
```

```
(defrule SkinAllergy5
   (DrySkin(dry_skin yes))
   (Flake(flake yes))
   (Rash(rash yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy has been diagnosed from
rash, dry skin and flake skin!! Allergies are pretty bad!
And the patient is having a really bad day.." crlf)
    (set-break GetNumbedMouth))


(defrule AllergikiRinitida1
   (RunningNose(running_nose yes))
   (ItsiEyes(itsi_eyes yes))
   (IntenseTears (intense_tears yes))
   (DifficultyInBreathing (difficulty_in_breathing yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from the patient having itsi eyes, intense tears,
difficulties in breathing and running nose- He is in
pretty bad shape.." crlf)
   (set-break GetBumpedLips))


(defrule AllergikiRinitida2
   (IntenseSneez (intense_sneez yes))
   (ItsiNose(itsi_nose yes))
   (RunningNose(running_nose yes))
   (ItsiEyes(itsi_eyes yes))

=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from intensive sneezing-atsou!!!!!!!!, itsi nose and
running nose" crlf)
   (set-break GetBumpedLips))


(defrule AllergikiRinitida3
   (ItsiNose (itsi_nose yes))
   (IntenseTears (intense_tears no))
   (Congest (congest no))
   (DifficultyInBreathing(difficulty_in_breathing yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from itsi nose and difficulty in breathing " crlf)
   (set-break GetRunningNose))
```

```
(defrule AllergikiRinitida4
    (ItsiEyes(itsi_eyes yes))
    (IntenseTears (intense_tears yes))
    (IntenseSneez(intense_sneez yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida diagnosed from itsi
eyes, intense tears and intense sneezing!" crlf)
   (set-break GetDifficultyInBreathing))


(defrule AllergikiRinitida5
   (IntenseSneez (intense_sneez yes))
   (ItsiNose(itsi_nose yes))
   (ItsiEyes(itsi_eyes yes))
   =>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from the intense sneezing, the itsi nose and eyes!The
horror! The horror! » crlf)
(set-break GetDifficultyInBreathing))
```

Για τον ίδιο λόγο που δημιουργήθηκε ο κανόνας διάγνωσης της ιλαράς, συντάχθηκαν και οι κανόνες διάγνωσης γρίπης. Ο στόχος εδώ είναι να αναγνωριστεί ποτέ ένας ασθενής έχει γρίπη και ποτέ αλλεργία καθώς αυτός ο διαχωρισμός είναι πολύ σημαντικός για την υγεία ενός ατόμου. Εάν κάποιος παρουσιάσει μέσης ή υψηλής έντασης πυρετό σε συνδυασμό με τον πονόλαιμο, τη καταρροή και το μπούκωμα μύτης διαπιστώνεται ότι έχει γρίπη ή κρυολόγημα και όχι αλλεργία. Τα συμπτώματα της γρίπης τείνουν να είναι παρόμοια με αυτά της αλλεργικής ρινίτιδας γι’ αυτό πρέπει το ιατρικό και κλινικό προσωπικό να είναι εξαιρετικά ενήμερο.

```
   (defrule Flu
   (SoreThroat (sore_throat yes))
   (Congest (congest yes))
   (Fever (level mild|high))
=>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from sore throat
and congest! mate is also having fever! go to bed!" crlf)
   (set-break GetDrySkin))


(defrule Flu2
   (Congest (congest yes))
   (RunningNose(running_nose yes))
   (Fever (level mild|high))
=>
```

```
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from fever,
running nose and congest!" crlf)
   (set-break GetBumpedLips))


(defrule Flu3
   (RunningNose (running_nose yes))
   (Fever (level mild|high))
   (SoreThroat (sore_throat yes))
=>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from fever,
running nose and sore throat" crlf)
   (set-break GetBumpedLips))


(defrule FoodAllergy1
   (BumpedLips (bumped_lips yes))
   (NumbedMouth(numbed_mouth yes))
   (IntenseTears(intense_tears yes))
=>
   (assert (Diagnosis (diagnosis foodallergy)))
   (printout t "Food allergy has been diagnosed from
mouth numbing, bumped lips and intense tears" crlf))
   (set-break GetVomit))


(defrule FoodAllergy2
   (NumbedMouth(numbed_mouth yes))
   (DrySkin(dry_skin yes))
   (Vomit(vomit yes))
=>
   (assert(Diagnosis (diagnosis foodallergy)))
   (printout t " Food allergy has been diagnosed  from
the dry skin, the numbed mouth and from the vomiting"
crlf)



(defrule FoodAllergy3
   (Flake(flake yes))
   (DrySkin(dry_skin yes))
   (NumbedMouth(numbed_mouth yes))
   (BumpedLips (bumped_lips yes))
=>
   (assert(Diagnosis(diagnosis foodallergy)))
   (printout t "Food Allergy has been diagnosed for the
unlucky patient from the flake and dry skin and the
numbed mouth and bumped lips" crlf))
   (set-break GetVomit))
```

```
(defrule FoodAllergy4
   (NumbedMouth(numbed_mouth yes))
   (BumpedLips (bumped_lips yes))
   (Vomit(vomit yes))
=>
   (assert(Diagnosis(diagnosis foodallergy)))
   (printout t "Food Allergy has been diagnosed for the
unlucky patient from the vomiting, the numbed mouth and
the bumped lips! He is really sick. " crlf))
```

3.2.4 Κανόνες θεραπείας : τέταρτο μέρος

Στο τέταρτο και τελευταίο μέρος του προγράμματος έγινε σύνθεση τεσσάρων κανόνων όπου ανάλογα με το είδος της πάθησης που εμφανίζει ο ασθενής προτείνουν μια αντίστοιχη θεραπεία υπενθυμίζοντας ότι οι πιθανές παθήσεις είναι δερματική αλλεργία, τροφική αλλεργία, αλλεργική ρινίτιδα, γρίπη και ιλαρά. Οι προτεινόμενες θεραπείες είναι:

- η χορήγηση αδρεναλίνης με τη μορφή ένεσης όταν διαγνωστεί ότι ο ασθενής έχει τροφική αλλεργία.
- η παροχή ιατρικών χαπιών στην περίπτωση που ο ασθενής έχει δερματική αλλεργία.
- η παροχή ιατρικών χαπιών στην περίπτωση που ο ασθενής έχει αλλεργική ρινίτιδα.
- η χορήγηση πενικιλίνης στην περίπτωση που ο ασθενής πάσχει από ιλαρά.
- και απλή ξεκούραση στην περίπτωση που πάσχει ο ασθενής από γρίπη.

Η δομή των κανόνων προτεινόμενης θεραπείας ανάλογα με το είδος της ασθένειας είναι ότι στο αριστερό μέρος τους (όπου για να ενεργοποιηθούν πρέπει να ισχύουν οι συνθήκες) βρίσκεται η διάγνωση που έχει γίνει από τους κανόνες διάγνωσης. Όταν πραγματοποιηθεί η αντίστοιχη διάγνωση για παράδειγμα, ότι ένας ασθενής πάσχει από ιλαρά, ενεργοποιείται ο κανόνας που στο αριστερό του μέρος περιέχει τη διάγνωση «ιλαρά» και έπειτα εισέρχεται στο πρότυπο της θεραπείας η προτεινόμενη θεραπεία εμφανίζοντας παράλληλα στην οθόνη το αντίστοιχο μήνυμα θεραπείας. Για το παράδειγμα που δόθηκε, στο πρότυπο “Treatment” θα εισαχθεί ως ιδιότητα η πενικιλίνη ενώ στην οθόνη θα εκτυπωθεί το μήνυμα:

```
 "Penicillin prescribed for the patient that has been
diagnosed with measles".
```

```
(defrule Penicillin
   (Diagnosis (diagnosis measles))
=>
   (assert (Treatment (treatment penicillin)))
   (printout t "Penicillin has been prescribed for the
patient that has been diagnosed with measles" crlf))


(defrule AllergyPills
   (Diagnosis (diagnosis skinallergy))
=>
   (assert (Treatment (treatment allergy_pills)))
   (printout t "Allergy pills have been prescribed for
the patient who has skin allergy" crlf))


(defrule BedRest
   (Diagnosis (diagnosis flu))
=>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest has been prescribed for the
tired and probably exhausted patient" crlf))


(defrule AllergyShot
   (Diagnosis(diagnosis foodallergy))
=>
   (assert(Treatment(treatment allergy_shot)))
   (printout t "Allergy shot with adrenaline has been
prescribed for the unlucky patient!" crlf))


(defrule AllergyPills2
   (Diagnosis (diagnosis allergikirinitida))
=>
   (assert (Treatment (treatment allergy_pills)))
   (printout t "Allergy pills- antihistaminic,
corticosteroid, aposymforitika- have been prescribed for
allergiki rinitida" crlf))
```

Είναι σημαντικό να δημιουργηθεί και ένας κανόνας που να βγάζει ως αποτέλεσμα ότι δεν είναι κάποια διάγνωση πιθανή με αυτά τα συμπτώματα και είναι προτιμότερο να συμβουλευτεί ο χρήστης έναν ειδικό και συγκεκριμένα έναν αλλεργιολόγο. Υπάρχουν περιπτώσεις από τις απαντήσεις (δηλαδή τον συνδυασμό) των ερωτήσεων να μην βγει πιθανή διάγνωση. Τότε στην οθόνη εκτυπώνεται το μήνυμα:

```
 "No diagnosis is possible – Please consult human expert.
Psychology is not my subroutines-I am a machine"

(defrule None
   (declare (salience -100))
   (not (diagnosis ?))
=>
   (printout t "No diagnosis is possible – Please consult
human expert. Psychology is not on my subroutines-I am a
machine!" crlf))
```

3.3 Δημιουργία έμπειρου συστήματος διάγνωσης αλλεργιών

Αφού έγινε αναλυτική περιγραφή του προβλήματος διάγνωσης αλλεργιών και εξηγήθηκαν οι κανόνες ερωτήσεων, διάγνωσης αλλεργιών και οι κανόνες θεραπείας, είναι ώρα να φανερωθεί ολοκληρωμένο το έμπειρο σύστημα και ο κώδικας στην γλώσσα CLIPS που το υλοποιεί. Το έμπειρο σύστημα που έχει δημιουργηθεί ονομάζεται "D.A.S" που είναι ακρωνύμιο των λέξεων "Diagnostic Allergy System" με το όνομα του να σημαίνει Διαγνωστικό Σύστημα Αλλεργιών. Με βάση όλα τα παραπάνω το τελικό πρόγραμμα που υλοποιήθηκε μέσω της γλώσσας CLIPS για τη διάγνωση και την αντιμετώπιση αλλεργιών είναι:

;Απλοποίηση του προγράμματος διάγνωσης αλλεργιών
;χρησιμοποιώντας ξεχωριστά αντικείμενα για κάθε σύμπτωμα ενώ κάθε
;φορά γίνεται μία και μόνο παρεμβολή για έναν ασθενή. Μετά ερωτάται
;το σύστημα για τον δεύτερο κλπ

```
(deftemplate Temperature (slot temperature))
(deftemplate Spots (slot spots))
(deftemplate Rash (slot rash))
(deftemplate SoreThroat  (slot sore_throat))
(deftemplate Innoculated  (slot innoculated))
(deftemplate DrySkin  (slot dry_skin))
(deftemplate ItsiNose  (slot itsi_nose))
(deftemplate Flake (slot flake))
(deftemplate RunningNose ( slot running_nose))
(deftemplate Congest  (slot congest))
(deftemplate IntenseSneez  (slot intense_sneez))
(deftemplate DifficultyInBreathing  (slot
difficulty_in_breathing))
(deftemplate NumbedMouth  (slot numbed_mouth))
(deftemplate BumpedLips  (slot bumped_lips))
(deftemplate ItsiEyes  (slot itsi_eyes))
(deftemplate IntenseTears (slot intense_tears))
(deftemplate Vomit (slot vomit))
(deftemplate Diagnosis   (slot diagnosis))
```

```
(deftemplate Treatment   (slot treatment))
(deftemplate Fever   (slot level))
```

;Οι πρώτοι κανόνες θα χρησιμοποιηθούν για να συλλέξουν τα ;συμπτώματα από τον χρηστή με την μορφή ερωτήσεων.
;Δεν υπάρχουν αρχικές συνθήκες στο αριστερό μέρος των κανόνων ;Get(ΌνομαΣυμπτώματος) που σημαίνει ότι θα τρέχουν ούτως ή αλλιώς.
;Οι δράσεις που βρίσκονται στο δεξί μέρος των κανόνων είναι η ;εκτύπωση της ερώτησης για το εάν ο ασθενής έχει το σύμπτωμα, η ;δέσμευση της συνάρτησης read με την μεταβλητή ?response και ;έπειτα η εισαγωγή ενός νέου γεγονότος με χρήση αυτής της τιμής.

```
(defrule GetTemperature
   =>
   (printout t "Please give the patient's temperature in
order to see if he has fever or not:  ")
   (bind ?response (read))
   (assert (Temperature (temperature ?response))))


(defrule GetSpots
   =>
   (printout t "Does the patient have spots on his face
or on his body? (yes or no):  ")
   (bind ?response (read))
   (assert (Spots (spots ?response))))
```

;Ερώτηση για συγκεκριμένη πληροφορία μόνο όταν είναι απαραίτητο. ;έχει νόημα να γίνει ερώτηση εάν ο ασθενής έχει κάνει εμβόλιο για την ;ιλαρά παρά μόνο εάν υπάρχει πιθανότητα να έχει ιλαρά.

```
(defrule GetInnoculated
   (Fever (level high))
   (Spots (spots yes))
   =>
   (printout t "Has the patient been innoculated for
measles or he was afraid the niddles? (yes or no):  ")
   (bind ?response (read))
   (assert (Innoculated (innoculated ?response))))


(defrule GetIntenseTears
=>
(printout t "Does the patient have too much tears-like he
is crying a lot for no reason? (yes or no): ")
(bind ?response (read))
(assert(IntenseTears(intense_tears ?response))))


(defrule GetRash
```

```
   =>
   (printout t "Does the patient have or feel a
rash(hour) generally? (yes or no):  ")
   (bind ?response (read))
   (assert (Rash (rash ?response))))


(defrule GetItsiEyes
=>
(printout t "Does the patient have the desire to scratch
his eyes because he feels them itsi(&scratsy) or hottie?
(yes or no):  ")
(bind ?response (read))
(assert(ItsiEyes (itsi_eyes ?response))))


(defrule GetSoreThroat
   =>
   (printout t "Does the patient have a sore throat? (yes
or no):  ")
   (bind ?response (read))
   (assert (SoreThroat (sore_throat ?response))))


(defrule GetItsiNose
=>
(printout t "Does the patient have itsi nose? (yes or no)
:  ")
(bind ?response (read))
(assert (ItsiNose(itsi_nose ?response))))


(defrule GetCongest
=>
(printout t "Does the patient has congest on his nose?
(yes or no) :  ")
(bind ?response (read))
(assert (Congest(congest ?response))))


(defrule GetDrySkin
=>
(printout t "Does the patient have dry skin? (yes or no):
")
(bind ?response (read))
(assert(DrySkin(dry_skin ?response))))


(defrule GetFlake
=>
```

```
(printout t "Does the patient have (corn)flake skin? (yes
or no) :  ")
(bind ?response (read))
(assert(Flake(flake ?response))))


(defrule GetNumbedMouth
=>
(printout t "Does the patient feel a numbing around his
mouth? (yes or no):  ")
(bind ?response (read))
(assert(NumbedMouth(numbed_mouth ?response))))


(defrule GetIntenseSneez
=>
(printout t "Does the patient have an intense sneez? (yes
or no):  ")
(bind ?response (read))
(assert(IntenseSneez(intense_sneez ?response))))


(defrule GetDifficultyInBreathing
=>
(printout t "Does the patient face difficulties in
breathing? (yes or no) :  ")
(bind ?response (read))
(assert(DifficultyInBreathing(difficulty_in_breathing
?response))))


(defrule GetRunningNose
=>
  (printout t "Does the patient have a running nose? (yes
or no) :  ")
  (bind ?response (read))
  (assert(RunningNose(running_nose ?response))))


(defrule GetBumpedLips
=>
(printout t "Does the patient have bumped lips? (yes or
no):  ")
(bind ?response (read))
(assert(BumpedLips (bumped_lips ?response))))


(defrule GetVomit
=>
(printout t "Does the patient have the feeling of
vomiting or he has already vomit? (yes or no):  ")
```

```
(bind ?response (read))
(assert(Vomit (vomit ?response))))
```

;Εδώ δίνονται οι κανόνες που δείχνουν εάν ο ασθενής έχει πυρετό και ;τι έντασης είναι μέσω της θερμοκρασίας.
;Αυτοί οι κανόνες βρίσκουν την θερμοκρασία του ασθενή και έπειτα την ;δεσμεύουν στο ?t. Στο επόμενο κομμάτι χρησιμοποιεί την συνάρτηση ;test για να εκτιμήσει τι πυρετό έντασης έχει ή δεν έχει ο ασθενής.

```
(defrule Fever1
   (Temperature (temperature ?t))
   (test (>=  ?t 38.5))
=>
   (assert (Fever (level high)))
   (printout t "Danger!!!! High fever diagnosed" crlf))


(defrule Fever2
   (Temperature (temperature ?t))
   (test (and (<  ?t  38.5) (>=  ?t  37)))
=>
   (assert (Fever (level mild)))
   (printout t "Mild fever diagnosed! Nothing to worry
about" crlf))


(defrule Fever3
   (Temperature (temperature ?t))
   (test (and (<  ?t  37) (>=  ?t 32)))
=>
   (assert(Fever(level normal)))
   (printout t " The patient has no signs of fever!
Everything is fine.." crlf))


(defrule Fever4
   (Temperature (temperature ?t))
   (test (<  ?t  32))
=>
   (assert(Fever(level paranormal)))
   (printout t "The patient has no signs of life! Is this
a joke?? Although I am an expert system, you shouldn't
make fun of me! I have feelings too!!" crlf))
```

;Κανόνες που καθορίζουν την διάγνωση με βάση τα συμπτώματα του ;ασθενή.
;Προσθήκη της Salience για να δοθεί στον κανόνα διάγνωσης της ;ιλαράς προτεραιότητα.

```
(defrule Measles
   (declare (salience 100))
   (Spots (spots yes))
   (Innoculated (innoculated no))
   (Fever (level high))
=>
   (assert (Diagnosis (diagnosis measles)))
   (printout t "Measles have been diagnosed for the
patient and this is not an allergy" crlf)
   (set-break GetIntenseTears))


(defrule SkinAllergy1
   (Rash (rash yes))
   (Spots(spots yes))
   (SoreThroat(sore_throat yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy unfortunately has been
diagnosed from rash, sore throat and from the appearance
of spots" crlf)
   (set-break GetItsiNose))


(defrule SkinAllergy2
   (Spots(spots yes))
   (DrySkin(dry_skin yes))
=>
   (assert (Diagnosis (diagnosis skinallergy)))
   (printout t "Skin allergy has been diagnosed from dry
skin and from the appearance of spots-I understand you
mate!" crlf)
   (set-break GetFlake))


(defrule SkinAllergy3
   (SoreThroat (sore_throat yes))
   (Rash(rash yes))
   (Flake(flake yes))
=>
    (assert (Diagnosis (diagnosis skinallergy)))
    (printout t "Skin allergy has been diagnosed from the
appearance of rash, flake and sore throating! Allergies
are pretty bad!" crlf)
    (set-break GetNumbedMouth))


(defrule SkinAllergy4
   (SoreThroat (sore_throat yes))
   (Rash(rash yes))
   (DrySkin (dry_skin yes))
```

```
=>
    (assert (Diagnosis (diagnosis skinallergy)))
    (printout t "Skin allergy has been diagnosed from the
appearance of rash, sore throat and dry skin! Allergies
are pretty bad!" crlf)
    (set-break GetFlake))


(defrule SkinAllergy5
   (DrySkin(dry_skin yes))
   (Flake(flake yes))
   (Rash(rash yes))
=>
    (assert (Diagnosis (diagnosis skinallergy)))
    (printout t "Skin allergy has been diagnosed from
rash, dry skin and flake skin!! Allergies are pretty bad!
And the patient is having a really bad day.." crlf)
    (set-break GetNumbedMouth))


(defrule AllergikiRinitida1
   (RunningNose(running_nose yes))
   (ItsiEyes(itsi_eyes yes))
   (IntenseTears (intense_tears yes))
   (DifficultyInBreathing (difficulty_in_breathing yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from the patient having itsi eyes, intense tears,
difficulties in breathing and running nose- He is in
pretty bad shape.." crlf)
   (set-break GetBumpedLips))


(defrule AllergikiRinitida2
   (IntenseSneez (intense_sneez yes))
   (ItsiNose(itsi_nose yes))
   (RunningNose(running_nose yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from intensive sneezing-atsou!!!!!!!, itsi nose and
running nose" crlf)
   (set-break GetBumpedLips))


(defrule AllergikiRinitida3
   (ItsiNose (itsi_nose yes))
   (IntenseTears (intense_tears no))
   (Congest (congest no))
   (DifficultyInBreathing(difficulty_in_breathing yes))
```

```
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from itsi nose and difficulty in breathing " crlf)
   (set-break GetRunningNose))


(defrule AllergikiRinitida4
    (ItsiEyes(itsi_eyes yes))
    (IntenseTears (intense_tears yes))
    (IntenseSneez(intense_sneez yes))
=>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from itsi eyes, intense tears and intense sneezing!"
crlf)
   (set-break GetDifficultyInBreathing))


(defrule AllergikiRinitida5
   (IntenseSneez (intense_sneez yes))
   (ItsiNose(itsi_nose yes))
   (ItsiEyes(itsi_eyes yes))
   =>
   (assert (Diagnosis (diagnosis allergikirinitida)))
   (printout t "Allergiki Rinitida has been diagnosed
from the intense sneezing, the itsi nose and eyes!The
horror! The horror! » crlf)
(set-break GetDifficultyInBreathing))


(defrule Flu
   (SoreThroat (sore_throat yes))
   (Congest (congest yes))
   (Fever (level mild|high))
=>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from sore throat
and congest! mate is also having fever! go to bed!" crlf)
   (set-break GetDrySkin))


(defrule Flu2
   (Congest (congest yes))
   (RunningNose(running_nose yes))
   (Fever (level mild|high))
=>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from fever,
running nose and congest!" crlf)
   (set-break GetBumpedLips))
```

```
(defrule Flu3
   (RunningNose (running_nose yes))
   (Fever (level mild|high))
   (SoreThroat (sore_throat yes))
=>
   (assert (Diagnosis (diagnosis flu)))
   (printout t "Flu has been diagnosed from fever,
running nose and sore throat" crlf)
   (set-break GetBumpedLips))


(defrule FoodAllergy1
   (BumpedLips (bumped_lips yes))
   (NumbedMouth(numbed_mouth yes))
   (IntenseTears(intense_tears yes))
=>
   (assert (Diagnosis (diagnosis foodallergy)))
   (printout t "Food allergy has been diagnosed from
mouth numbing, bumped lips and intense tears" crlf)
   (set-break GetVomit))


(defrule FoodAllergy2
   (NumbedMouth(numbed_mouth yes))
   (DrySkin(dry_skin yes))
   (Vomit(vomit yes))
=>
   (assert(Diagnosis (diagnosis foodallergy)))
   (printout t "Food allergy has been diagnosed  from the
dry skin, the numbed mouth and from the vomiting" crlf))


(defrule FoodAllergy3
   (Flake(flake yes))
   (DrySkin(dry_skin yes))
   (NumbedMouth(numbed_mouth yes))
   (BumpedLips (bumped_lips yes))
=>
   (assert(Diagnosis(diagnosis foodallergy)))
   (printout t "Food Allergy has been diagnosed for the
unlucky patient from the flake and dry skin and the
numbed mouth and bumped lips" crlf)
   (set-break GetVomit))


(defrule FoodAllergy4
   (NumbedMouth(numbed_mouth yes))
   (BumpedLips (bumped_lips yes))
   (Vomit(vomit yes))
```

```
=>
   (assert(Diagnosis(diagnosis foodallergy)))
   (printout t "Food Allergy has been diagnosed for the
unlucky patient from the vomiting, the numbed mouth and
the bumped lips! He is really sick.. " crlf))
```

;Κανόνες που προτείνουν θεραπείες με βάση την διάγνωση που ;πραγματοποιήθηκε.

```
(defrule Penicillin
   (Diagnosis (diagnosis measles))
=>
   (assert (Treatment (treatment penicillin)))
   (printout t "Penicillin has been prescribed for the
patient that has been diagnosed with measles" crlf))


(defrule AllergyPills
   (Diagnosis (diagnosis skinallergy))
=>
   (assert (Treatment (treatment allergy_pills)))
   (printout t "Allergy pills have been prescribed for
the patient who has skin allergy" crlf))


(defrule BedRest
   (Diagnosis (diagnosis flu))
=>
   (assert (Treatment (treatment bed_rest)))
   (printout t "Bed rest has been prescribed for the
tired and probably exhausted patient" crlf))


(defrule AllergyShot
   (Diagnosis(diagnosis foodallergy))
=>
   (assert(Treatment(treatment allergy_shot)))
   (printout t "Allergy shot with adrenaline has been
prescribed for the unlucky patient!" crlf))


(defrule AllergyPills2
   (Diagnosis (diagnosis allergikirinitida))
=>
   (assert (Treatment (treatment allergy_pills)))
   (printout t "Allergy pills-antihistaminic,
corticosteroid, aposymforitika- have been prescribed for
allergiki rinitida" crlf))
```

```
(defrule None
   (declare (salience -100))
   (not (diagnosis ?))
=>
   (printout t "No diagnosis is possible – Please consult
human expert. Psychology is not on my subroutines-I am a
machine!" crlf))
```

3.4 Μία δεύτερη προσέγγιση

Για την επίλυση του προβλήματος όπως προαναφέρθηκε (3.2.1) δημιουργήθηκαν ερωτήσεις μέσω κανόνων όπου ανάλογα με τις απαντήσεις που δίνονται πραγματοποιείται κατάληξη σε κάποιο είδος αλλεργίας και σε μια προτεινόμενη αντίστοιχη θεραπεία ή όχι. Ωστόσο αξίζει να αναφερθεί ότι το πρόβλημα μπορεί να επιλυθεί χρησιμοποιώντας άλλη προσέγγιση εκτός από αυτήν που ακολουθήθηκε.

Η εισαγωγή γεγονότων χρησιμοποιήθηκε ως εναλλακτικός τρόπος δημιουργίας του έμπειρου συστήματος διάγνωσης αλλεργιών. Πιο συγκεκριμένα τα συμπτώματα που ενδέχεται να παρουσιάσει ο ασθενής, εισέρχονται ως γεγονότα και πιο αναλυτικά ως ιδιότητες ενός προτύπου που καλείται «Ασθενής». Με βάση αυτόν τον συνδυασμό γεγονότων που εισέρχεται στο σύστημα και έχοντας καθορίσει τους αντιστοίχους κανόνες διάγνωσης και θεραπείας, εάν πληκτρολογηθεί (reset) και έπειτα (run), εμφανίζεται η πιθανή διάγνωση και θεραπεία που προτείνει το σύστημα. Το σύστημα που δημιουργείται με αυτόν τον εναλλακτικό τρόπο αποτελείται και πάλι από τέσσερα μέρη.

Σχεδιάζεται λοιπόν ένα πρότυπο γεγονότος για τον ασθενή το οποίο περιέχει ως ιδιότητες το όνομα του ασθενή και τα πιθανά συμπτώματα (γρίπης, αλλεργίας, ιλαράς). Επίσης δημιουργούνται πρότυπα για τη διάγνωση και για τη θεραπεία.

```
(deftemplate Patient
   (slot name)
   (slot fever)
   (slot spots)
   (slot rash)
   (slot sore_throat)
   (slot innoculated)
   (slot dry_skin)
   (slot running_nose)
   (slot itsi_eyes))

(deftemplate Diagnosis
   (slot name)
   (slot diagnosis))

(deftemplate Treatment
   (slot name)
```

```
   (slot treatment))
```

Οι κανόνες διάγνωσης αποτελούνται στο αριστερό τους μέρος από το πρότυπο του ασθενή με τιμές στις ιδιότητες του. Αυτός ο συνδυασμός τιμών είναι που καθορίζει τη διάγνωση για τον ασθενή. Συντάσσεται ένας κανόνας διάγνωσης ιλαράς, τέσσερις για τη διάγνωση αλλεργιών και ένας ακόμη για τη διάγνωση γρίπης. Από την στιγμή που δοθεί ένας συνδυασμός που βρίσκεται στο αριστερό μέρος ενός κανόνα τότε πραγματοποιείται εισαγωγή στο πρότυπο της διάγνωσης της ασθένειας που αναφέρει ο κανόνας και εκτυπώνεται στην οθόνη το αντίστοιχο μήνυμα διάγνωσης αυτής. Η “?n” είναι μια μεταβλητή που δεσμεύεται με την άξια που αποδίδεται στην ιδιότητα (name) του πρότυπου του ασθενή. Αυτή η τιμή χρησιμοποιείται και στον υπόλοιπο κανόνα και στην εντολή (assert) και στην εντολή (printout).

```
(defrule Measles
   (declare (salience 100))
   (Patient (name ?n)(fever high) (spots yes)
(innoculated no))
=>
   (assert (Diagnosis (name ?n) (diagnosis measles)))
   (printout t "Measles have been diagnosed for " ?n
crlf))


(defrule Allergy1
   (Patient (name ?n) (spots yes)(rash yes))
=>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
   (printout t "Allergy has been diagnosed from spots and
rash for" ?n crlf))


(defrule Allergy2
   (Patient (name ?n)(itsi_eyes yes)(running_nose
yes)(dry_skin yes))
=>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
   (printout t "Allergy has been diagnosed from rash, dry
skin, running nose and itsi eyes for " ?n crlf))


(defrule Allergy3
   (Patient (name ?n) (spots yes)(running_nose
yes)(sore_throat yes))
=>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
   (printout t "Allergy has been diagnosed from rash, dry
skin, running nose and itsi eyes for " ?n crlf))
```

```
(defrule Allergy4
   (Patient (name ?n) (dry_skin yes)(spots
yes)(sore_throat no))
=>
   (assert (Diagnosis (name ?n) (diagnosis allergy)))
   (printout t "Allergy has been diagnosed from dry skin
and spots " ?n crlf))


(defrule Flu
   (Patient (name ?n) (sore_throat yes) (fever mild |
high))
=>
   (assert (Diagnosis (name ?n) (diagnosis flu)))
   (printout t "Flu has been diagnosed for " ?n crlf))
```

Αφού σχεδιαστούν οι κανόνες διάγνωσης θα πρέπει να δημιουργηθούν και οι κανόνες θεραπείας. Η δομή τους είναι παρόμοια με αυτή των κανόνων διάγνωσης. Από την στιγμή που θα πραγματοποιηθεί διάγνωση και στο αριστερό μέρος του κανόνα βρίσκεται η διάγνωση που έγινε τότε γίνεται εισαγωγή στο πρότυπο της θεραπείας ενός γεγονότος με την ονομασία της θεραπείας για την συγκεκριμένη πάθηση ενώ παράλληλα εκτυπώνεται και στην οθόνη το αντίστοιχο μήνυμα πρότασης θεραπείας.

```
(defrule Penicillin
   (Diagnosis(name ?n) (diagnosis measles))
=>
   (assert (Treatment (name ?n) (treatment penicillin)))
   (printout t "Penicillin has been prescribed for " ?n
crlf))


(defrule Allergy_pills
   (Diagnosis (name ?n) (diagnosis allergy))
=>
   (assert (Treatment (name ?n) (treatment
allergy_shot)))
   (printout t "Allergy shot has been prescribed
unfortunately for the unlucky patient whose name is " ?n
crlf))


(defrule Bed_rest
   (Diagnosis (name ?n) (diagnosis flu))
=>
   (assert (Treatment (name ?n) (treatment bed_rest)))
   (printout t "Bed rest has been prescribed for " ?n
crlf))
```

```
(defrule None
   (declare (salience -100))
   (not (diagnosis ?))
=>
   (printout t "No diagnosis possible – Please consult
human expert. Psychology is on my subroutines. I am a
machine(Arni quote from T3!..)" crlf))
```

Τα γεγονότα δημιουργούνται εδώ με την εντολή (deffacts). Για παράδειγμα:

```
(deffacts Symptoms
   (Patient (name "MeMyselfAndI")
            (fever low)
            (spots yes)
            (rash no)
            (sore_throat no)
            (innoculated no)))
```

Με στόχο να εισέλθουν αυτά τα ομαδοποιημένα γεγονότα στη CLIPS πρέπει να πληκτρολογηθεί (reset). Έπειτα πατώντας (run), με βάση τον συνδυασμό συμπτωμάτων που δόθηκε ως γεγονότα, το σύστημα ψάχνει να βρει εάν υπάρχει κάποιος κανόνας διάγνωσης που να περιέχει αυτό τον συνδυασμό συμπτωμάτων και να εμφανίσει ως μήνυμα την αντίστοιχη διάγνωση και θεραπεία που ενδεχομένως να υπάρχει.

3.5 Σύνοψη

Σε αυτό το κεφαλαίο δόθηκε ο ορισμός της αλλεργίας και αναλύθηκαν τα είδη των αλλεργιών και τα αντίστοιχα συμπτώματά τους. Ακολούθησε περιγραφή του προβλήματος το οποίο αφορά τη δημιουργία ενός έμπειρου συστήματος διάγνωσης αλλεργιών γραμμένο στη γλώσσα CLIPS. Στην πορεία πραγματοποιήθηκε υλοποίηση του συστήματος μέσω κανόνων ερωτήσεων και πρότυπων γεγονότων όντας ο κύριος τρόπος επίλυσης του προβλήματος, ενώ παρουσιάστηκε και μία εναλλακτική προσέγγιση μέσω εισαγωγής γεγονότων για οποίον υποψήφιο χρήστη θεωρεί ότι ο πρώτος τρόπος δεν είναι ευέλικτος και αποδοτικός .

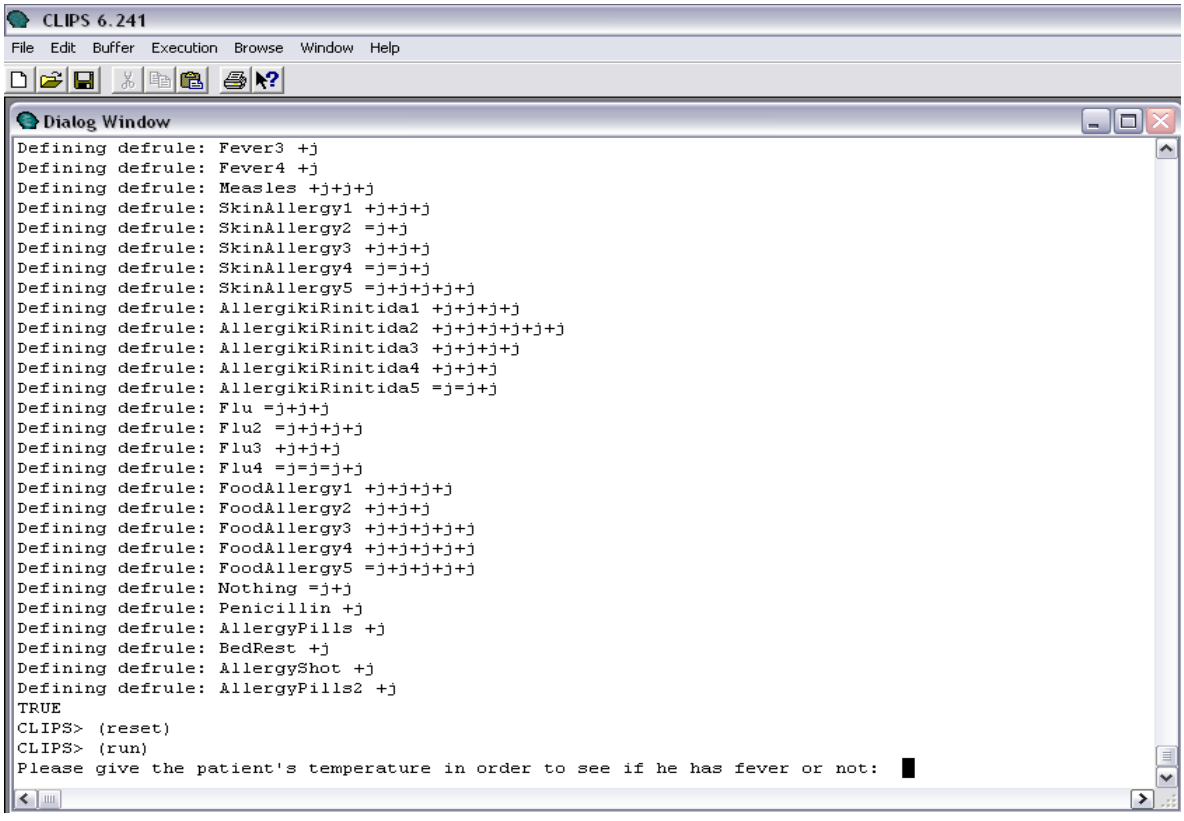
Κεφαλαίο 4: Αποτελέσματα- Συζήτηση- Συμπεράσματα

4.1 Αποτελέσματα

Μετά την ολοκληρωμένη περιγραφή και παρουσίαση του προβλήματος στο κεφάλαιο 3, θα δειχτεί ότι το “D.A.S” καταλήγει σε διάγνωση για το εάν ο ασθενής έχει αλλεργία ή όχι προτείνοντας και την αντίστοιχη θεραπεία. Για τη πλήρη κατανόηση της λειτουργίας του θα δειχθούν κάποιες περιπτώσεις ασθενών.

4.1.1 Πρώτη περίπτωση : διάγνωση ιλαράς

Έστω ένας ασθενής που παρουσιάζει υψηλό πυρετό και εξανθήματα ενώ δεν έχει κάνει εμβόλιο για την ιλαρά. Πληκτρολογώντας την εντολή (reset), καθαρίζεται η ατζέντα από κανόνες και αρχικοποιείται το σύστημα. Έπειτα μέσω της εντολής (run) τρέχεται το πρόγραμμα με τους κανόνες οι οποίοι έχουν φορτωθεί. Το “D.A.S” τότε ζητά να δοθεί την θερμοκρασία του ασθενή (Εικόνα 4.1.1.1).

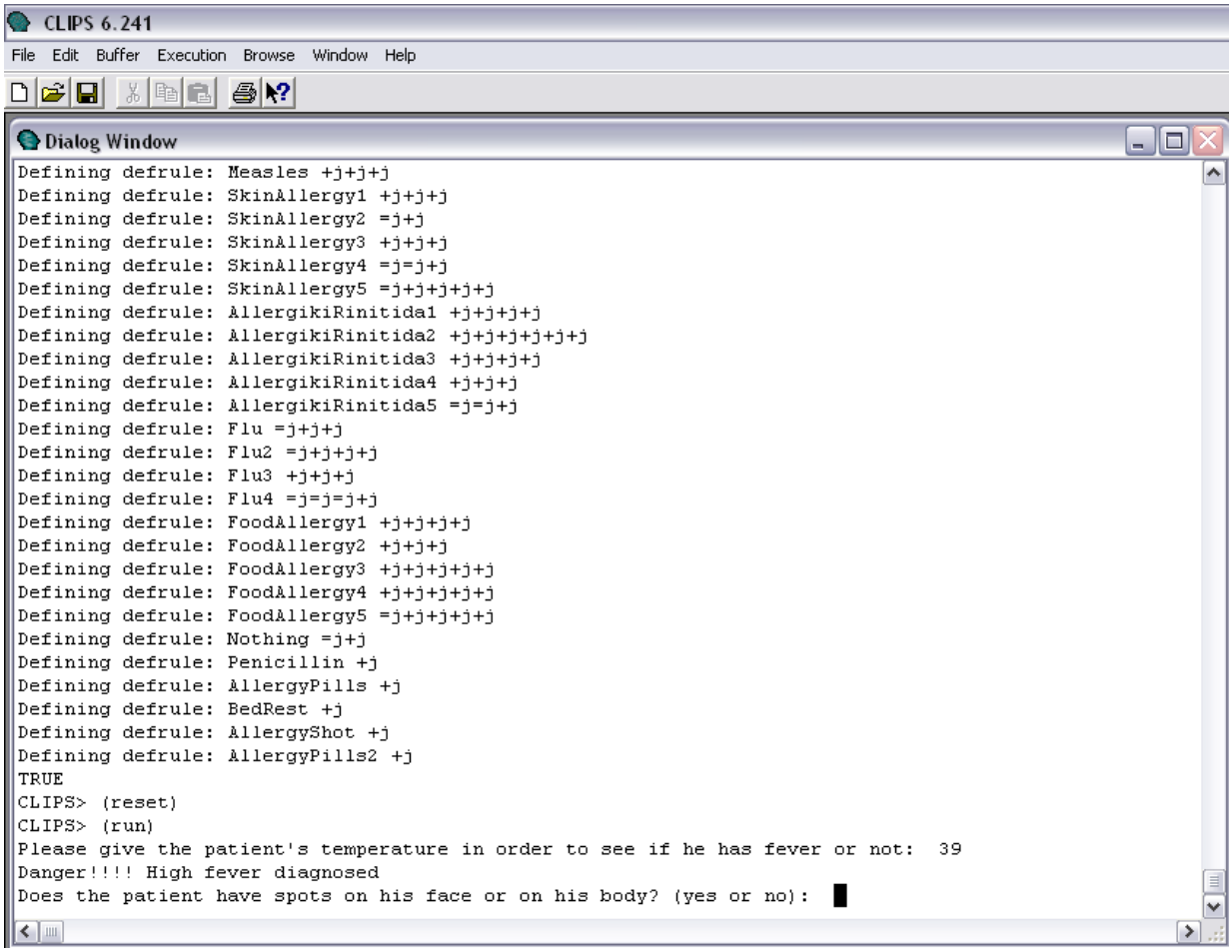


Εικόνα 4.1.1.1 Το έμπειρο σύστημα ζητά από τον χρήστη να εισάγει την θερμοκρασία του ασθενή.

Εφόσον ο ασθενής έχει υψηλό πυρετό πληκτρολογείται ένας αριθμός πάνω από το 38,5 °C για παράδειγμα 39 °C. Πραγματοποιείται κατά συνέπεια διάγνωση λέγοντας ότι ο ασθενής έχει υψηλό πυρετό και εμφανίζεται το μήνυμα:

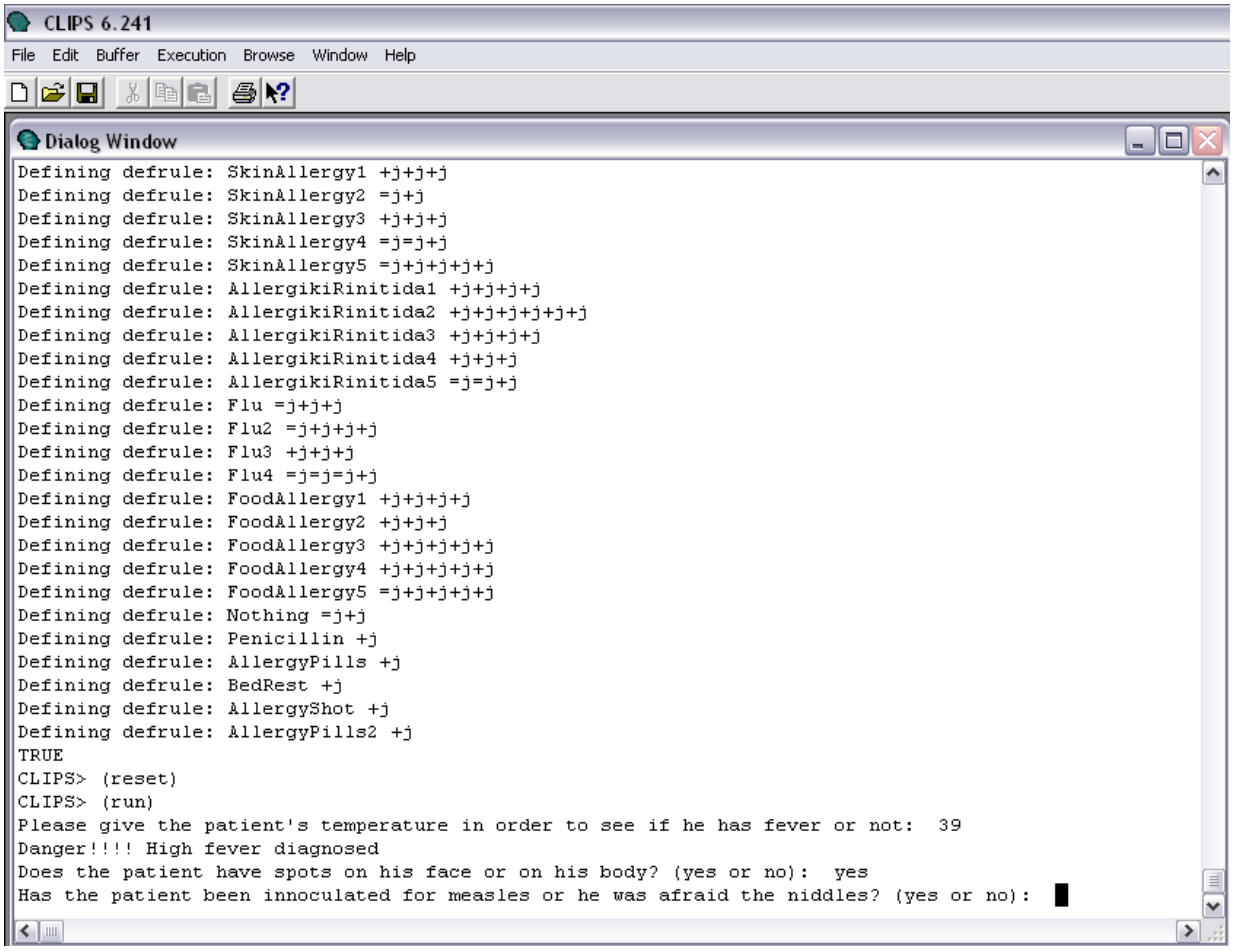
`"Danger!!!High fever Diagnosed"`

μαζί με την επόμενη ερώτηση για τα εξανθήματα (Εικόνα 4.1.1.2).



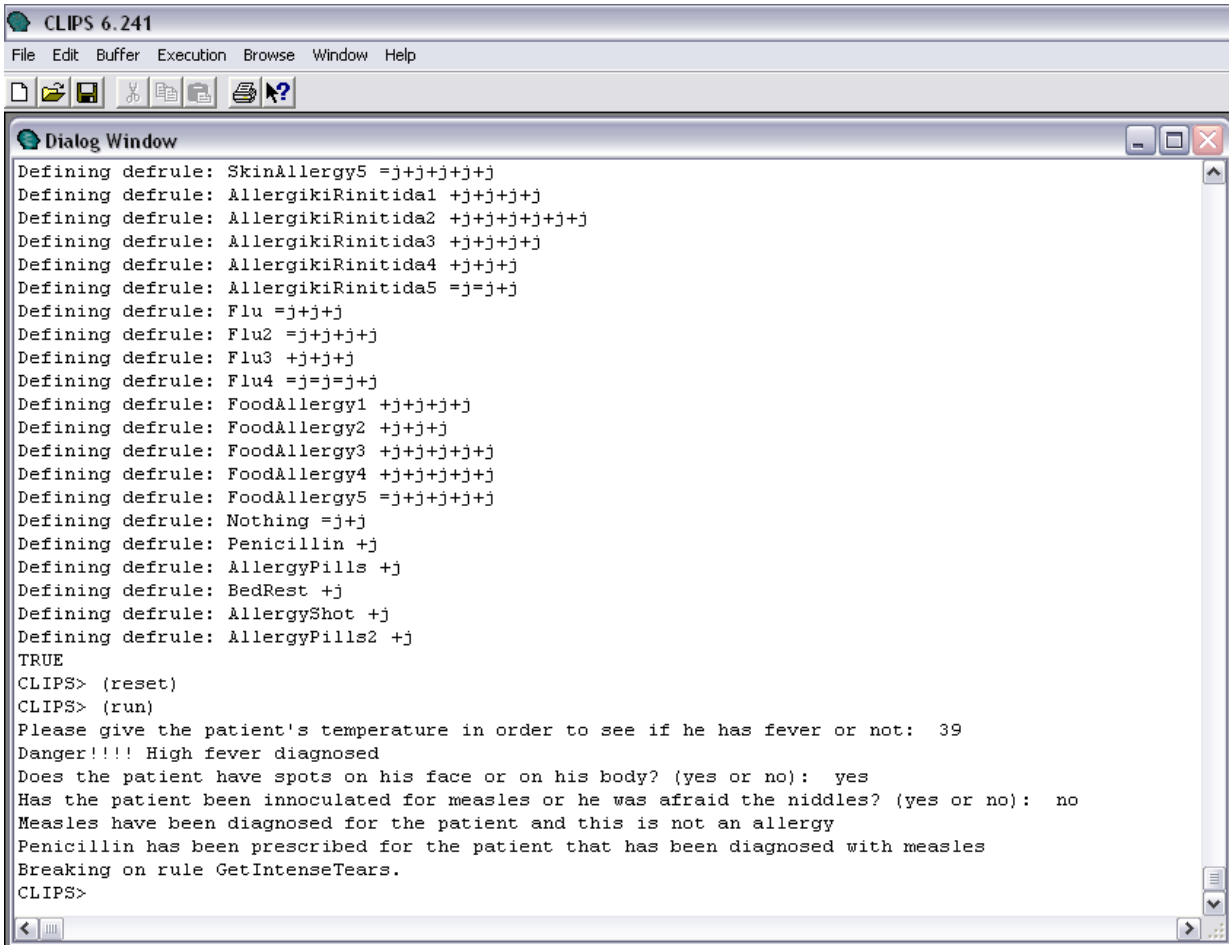
Εικόνα 4.1.1.2 Το έμπειρο σύστημα ρωτά τον χρήστη εάν ο ασθενής εμφανίζει εξανθήματα στο πρόσωπο ή στο σώμα του.

Γνωρίζοντας ότι ο ασθενής όντως έχει εξανθήματα η απάντηση είναι “yes”. Πατώντας “yes”, η επόμενη ερώτηση που προκύπτει αναφέρει εάν ο ασθενής έχει κάνει εμβόλιο για την ιλαρά (Εικόνα 4.1.1.3). Να επισημανθεί στο σημείο αυτό ότι έχοντας ο ασθενής πυρετό και εξανθήματα, αμέσως η επόμενη ερώτηση που γίνεται αφορά το εμβόλιο κατά της ιλαράς, ακριβώς γιατί έχει δοθεί προτεραιότητα σε αυτόν τον κανόνα διάγνωσης προκείμενου γρήγορα εάν απαντηθεί καταφατικά η ερώτηση, να απορριφθεί το ενδεχόμενο της ιλαράς.



Εικόνα 4.1.1.3 Το έμπειρο σύστημα ρωτά τον χρήστη εάν ο ασθενής έχει κάνει εμβόλιο για την ασθένεια της ιλαράς.

Εφόσον είναι γνωστό ότι δεν έχει κάνει, η απάντηση είναι αρνητική πληκτρολογώντας “no” και το σύστημα αναφέρει τη πιθανή διάγνωση και το τι προτείνει για την πάθηση που έχει ο ασθενής. Όπως παρατηρείται στην συγκεκριμένη περίπτωση, ο ασθενής πάσχει από ιλαρά και ο κατάλληλος τρόπος αντιμετώπισης της ασθένειας αυτής είναι να γίνει χορήγηση πενικιλίνης. Να σημειωθεί ότι από τη στιγμή που το έμπειρο σύστημα έκανε διάγνωση και πρότεινε θεραπεία, οι υπόλοιπες ερωτήσεις δεν πραγματοποιούνται (Εικόνα 4.1.1.4).



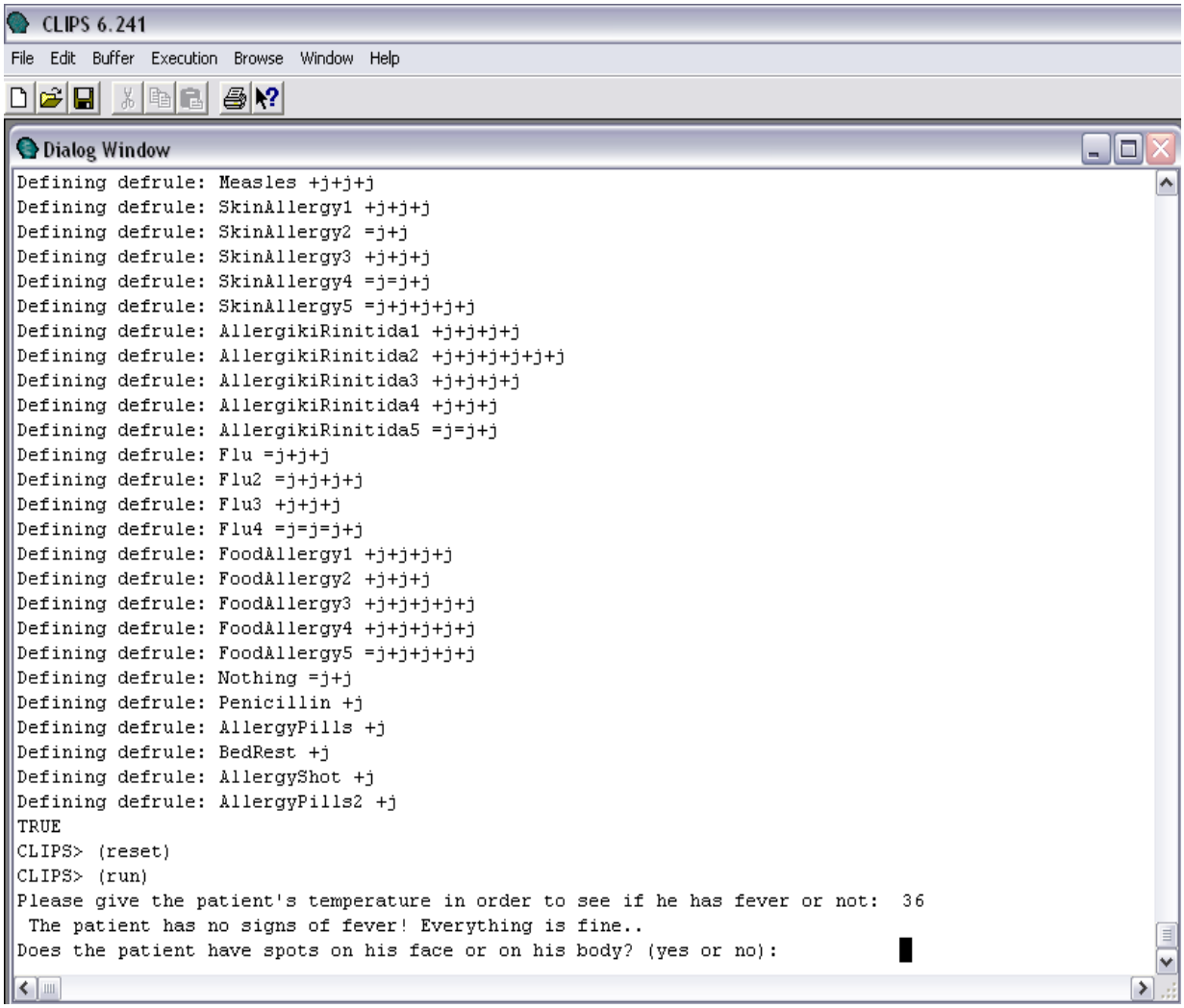
Εικόνα 4.1.1.4 Διάγνωση του έμπειρου συστήματος ότι ο ασθενής με συμπτώματα εμφάνισης υψηλού πυρετού και εξανθημάτων χωρίς να έχει εμβολιαστεί για την ασθένεια της ιλαράς πάσχει από ιλαρά και προτείνει να του χορηγηθεί πενικιλίνη.

4.1.2 Δεύτερη περίπτωση: διάγνωση αλλεργικής ρινίτιδας

Έστω ένας ασθενής με έντονο φτέρνισμα, φαγούρα στη μύτη χωρίς καταρροή και κνησμό ματιών. Ξαναπατώντας (reset) και έπειτα (run), το “D.A.S” ζητά την θερμοκρασία του ασθενή. Από τη στιγμή που δεν υπάρχει κάποιο στοιχείο ή ένδειξη ότι παρουσιάζει ο ασθενής πυρετό θεωρείται ότι η θερμοκρασία του σώματος του είναι φυσιολογική και κατά αυτόν τον τρόπο πληκτρολογείται μία τιμή μεταξύ 37 °C και 32 °C (όπως έχει οριστεί στον αντίστοιχο κανόνα fever3). Έστω 36 °C. Παρατηρείται ότι το σύστημα αναφέρει ότι τα πάντα είναι μια χαρά και εμφανίζεται το εξής μήνυμα:

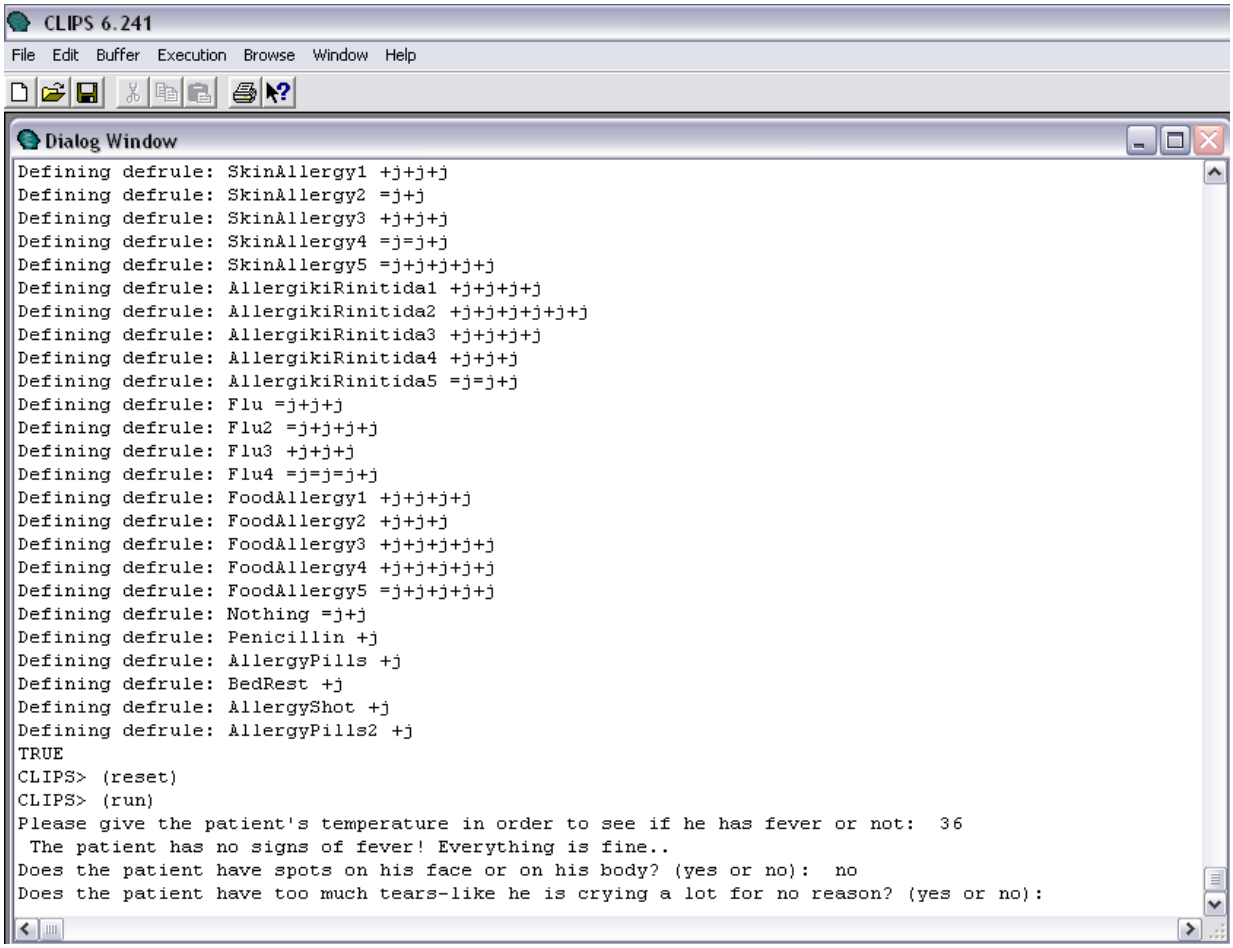
`“The patient has no sings of fever! Everything is fine..”`

δηλώνοντας ότι ο ασθενής δεν έχει πυρετό (Εικόνα 4.1.2.1).



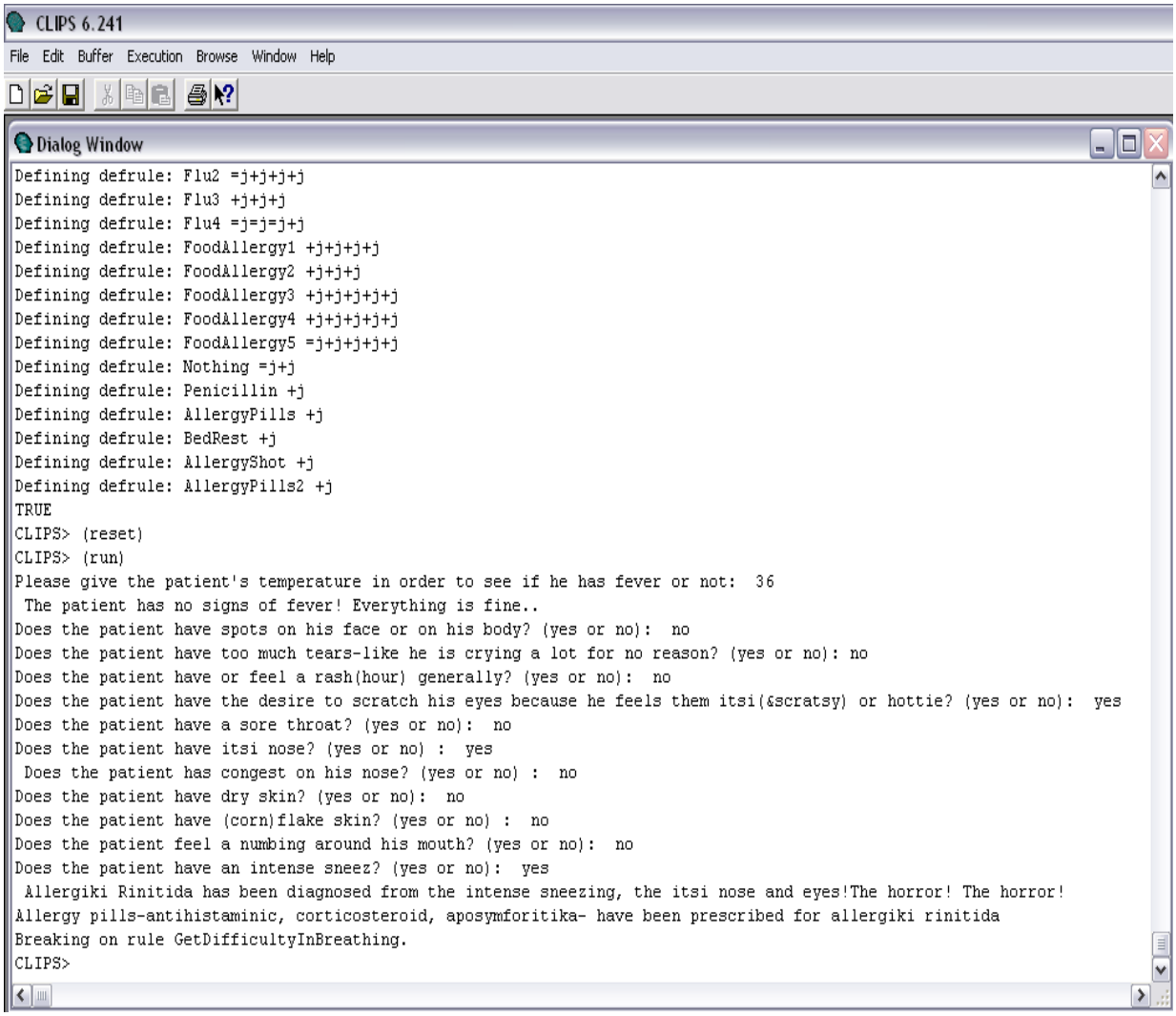
Εικόνα 4.1.2.1 Το έμπειρο σύστημα «D.A.S» ρωτά τον χρήστη εάν ο ασθενής εμφανίζει εξανθήματα εφόσον προηγουμένως έχει ζητήσει την θερμοκρασία του ασθενή για να κρίνει εάν έχει πυρετό.

Η επόμενη ερώτηση αφορά την εμφάνιση εξανθημάτων στον ασθενή. Απαντώντας αρνητικά με ένα “no” και διακρίνεται ότι στην οθόνη εμφανίζεται η επόμενη ερώτηση για το εάν ο ασθενής έχει δακρύρροια (Εικόνα 4.1.2.2).



Εικόνα 4.1.2.2 Το έμπειρο σύστημα ρωτά τον χρήστη εάν ο ασθενής εμφανίζει δακρύρροια.

Απαντώντας και πάλι αρνητικά, πραγματοποιείται συνεχεία όσον αφορά τον τρόπο απάντησης των ερωτήσεων συμπτωμάτων που κάνει το σύστημα είτε αρνητικά είτε καταφατικά (με βάση τα στοιχεία που υπάρχουν για τον ασθενή). Στην Εικόνα 4.1.2.3 παρατηρείται τελικά ότι η διάγνωση πραγματοποιήθηκε από την στιγμή που απαντήθηκε η ερώτηση για το εάν ο ασθενής έχει ή όχι καταρροή μύτης.

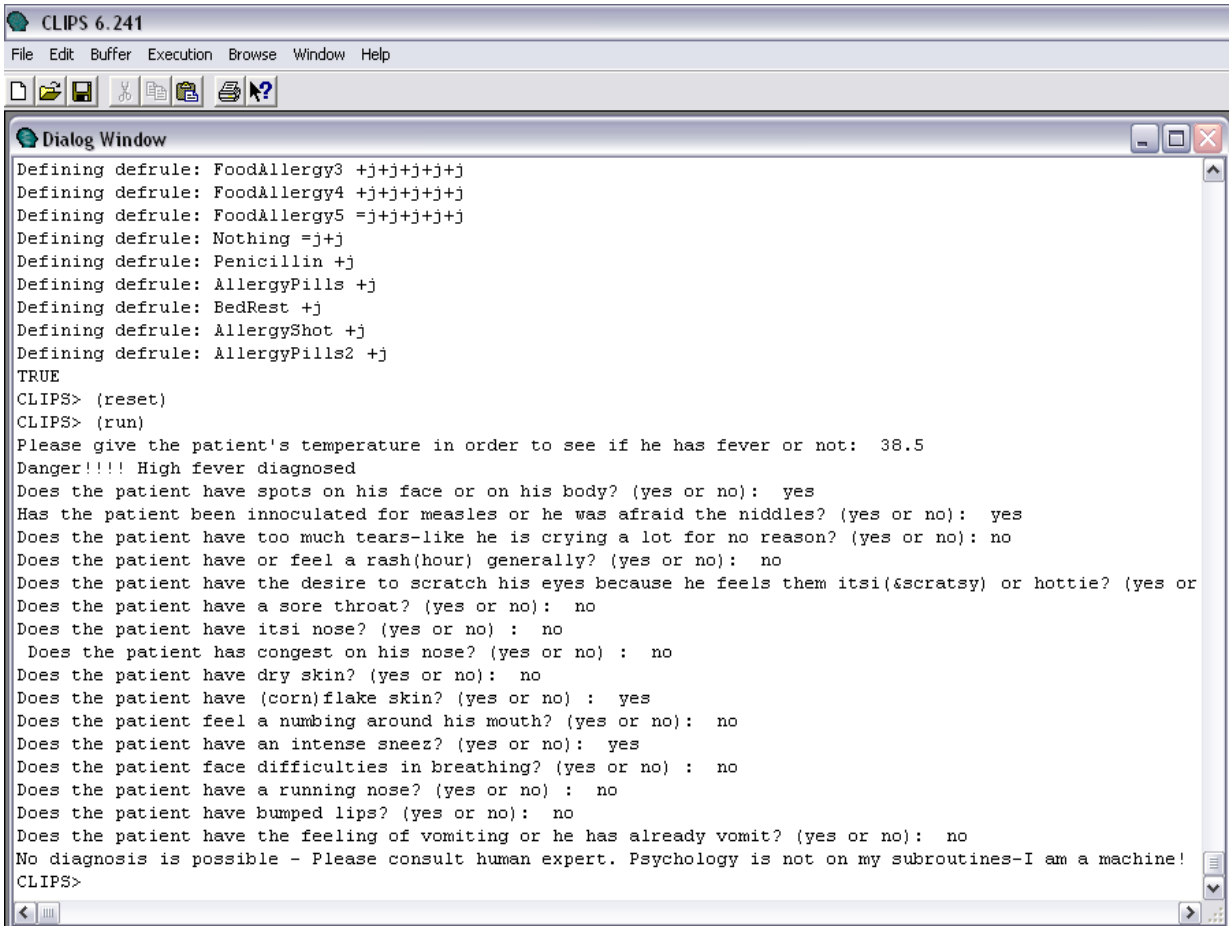


Εικόνα 4.1.2.3 Διάγνωση του έμπειρου συστήματος «D.A.S» για τον ασθενή που εμφανίζει συμπτώματα έντονου φτερνίσματος, κνησμού μύτης και ματιών ότι πάσχει από αλλεργική ρινίτιδα και πρόταση για χορήγηση αντιισταμινικών χαπιών.

Σύμφωνα με την διάγνωση του έμπειρου συστήματος “D.A.S”, ο ασθενής πάσχει από αλλεργική ρινίτιδα εξαιτίας του έντονου φτερνίσματος, του κνησμού ματιών και μύτης ενώ η κατάλληλη θεραπεία είναι η χορήγηση αντιισταμινικών και κορτικοστεροειδών χαπιών.

4.1.3 Τρίτη περίπτωση: δεν υπάρχει πιθανή διάγνωση

Έστω ότι ο ασθενής εμφανίζει συμπτώματα υψηλού πυρετού, έχει εξανθήματα, έχει κάνει ωστόσο εμβόλιο για την ιλαρά, φτερνίζεται έντονα και παρουσιάζει ξεφλούδισμα δέρματος. Ξεκινώντας πάλι το σύστημα με τον ίδιο τρόπο πατώντας (reset) και (run), δίνεται πάλι υψηλή θερμοκρασία για τον ασθενή (38,5 °C) και οι ερωτήσεις απαντώνται με βάση τα στοιχεία που υπάρχουν. Το “D.A.S” αναφέρει ότι με τα συγκεκριμένα συμπτώματα που έχει ο ασθενής δεν μπόρεσε να βγάλει κάποια πιθανή διάγνωση και προτείνει τον χρήστη να συμβουλευτεί κάποιον ειδικό ιατρό όπως αλλεργιολόγο, εμφανίζοντας στην οθόνη τα αντίστοιχα μηνύματα με μία ελαφριά δόση χιούμορ (Εικόνα 4.1.3.1).



Εικόνα 4.1.3.1 Το έμπειρο σύστημα «D.A.S» δεν κατέληξε σε κάποια συγκεκριμένη διάγνωση γι’ αυτήν την περίπτωση ασθενούς και προτείνει να συμβουλευτεί ο χρήστης έναν εμπειρογνώμονα.

4.2 Αποτελέσματα δεύτερης προσέγγισης

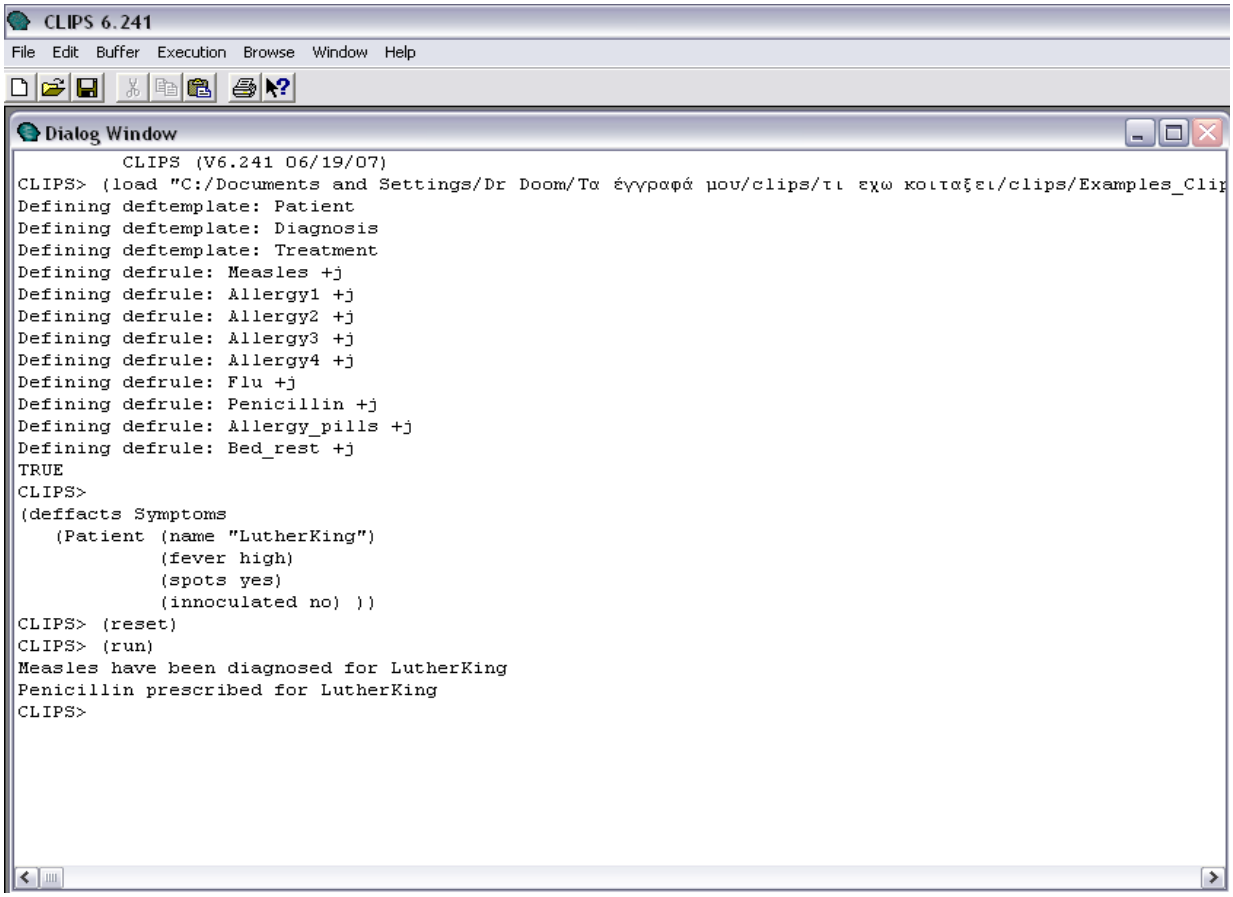
Το έμπειρο σύστημα που δημιουργήθηκε με αυτόν τον τρόπο-που θεωρείται εναλλακτική προσέγγιση για όποιον θεωρεί ότι ο πρώτος τρόπος είναι πολύπλοκος και χρονοβόρος, ονομάζεται “D.A.S.2”. Ομοίως θα δειχτούν δυο περιπτώσεις προκειμένου να παρατηρηθεί πως δουλεύει.

4.2.1 Πρώτη περίπτωση: διάγνωση ιλαράς

Έστω ότι έχουμε έναν ασθενή που λέγεται LutherKing. Ο LutherKing έχει υψηλό πυρετό, εμφανίζει εξανθήματα ενώ δεν έχει εμβολιαστεί κατά της ιλαράς. Αυτά τα στοιχεία περνιούνται ως γεγονότα στην CLIPS με την εντολή (deffacts).Φτιάχνουμε λοιπόν την εξής λίστα γεγονότων:

```
(deffacts Symptoms
   (Patient (name “LutherKing”)
            (fever high)
            (spots yes)
            (innoculated no)))
```

Για να περαστούν όμως τα γεγονότα αυτά στη CLIPS θα πρέπει να πληκτρολογηθεί (reset) αφού αυτό είναι το μειονέκτημα της εντολής (deffacts) για την εισαγωγή μιας ομάδας γεγονότων. Μόλις πληκτρολογηθεί η εντολή (reset), τα γεγονότα εισέρχονται στο σύστημα. Για την εμφάνιση της πιθανής διάγνωσης και της προτεινόμενης θεραπείας που δίνει το “D.A.S.2” πληκτρολογείται η εντολή (run) ώστε να τρεχθεί το πρόγραμμα. Τότε εμφανίζεται η διάγνωση και η προτεινόμενη θεραπεία εάν υπάρχει. Όπως διαπιστώνεται και στην Εικόνα 4.2.1.1 το “D.A.S.2” διέγνωσε ότι ο LutherKing πάσχει από ιλαρά και πρέπει να του χορηγηθεί πενικιλίνη.



Εικόνα 4.2.1.1 Το έμπειρο σύστημα διέγνωσε ότι ο συγκεκριμένος ασθενής που εμφανίζει συμπτώματα υψηλού πυρετού και εξανθήματα χωρίς να έχει εμβολιαστεί κατά της ιλαράς πάσχει από ιλαρά και η προτεινόμενη θεραπεία είναι η χορήγηση πενικιλίνης.

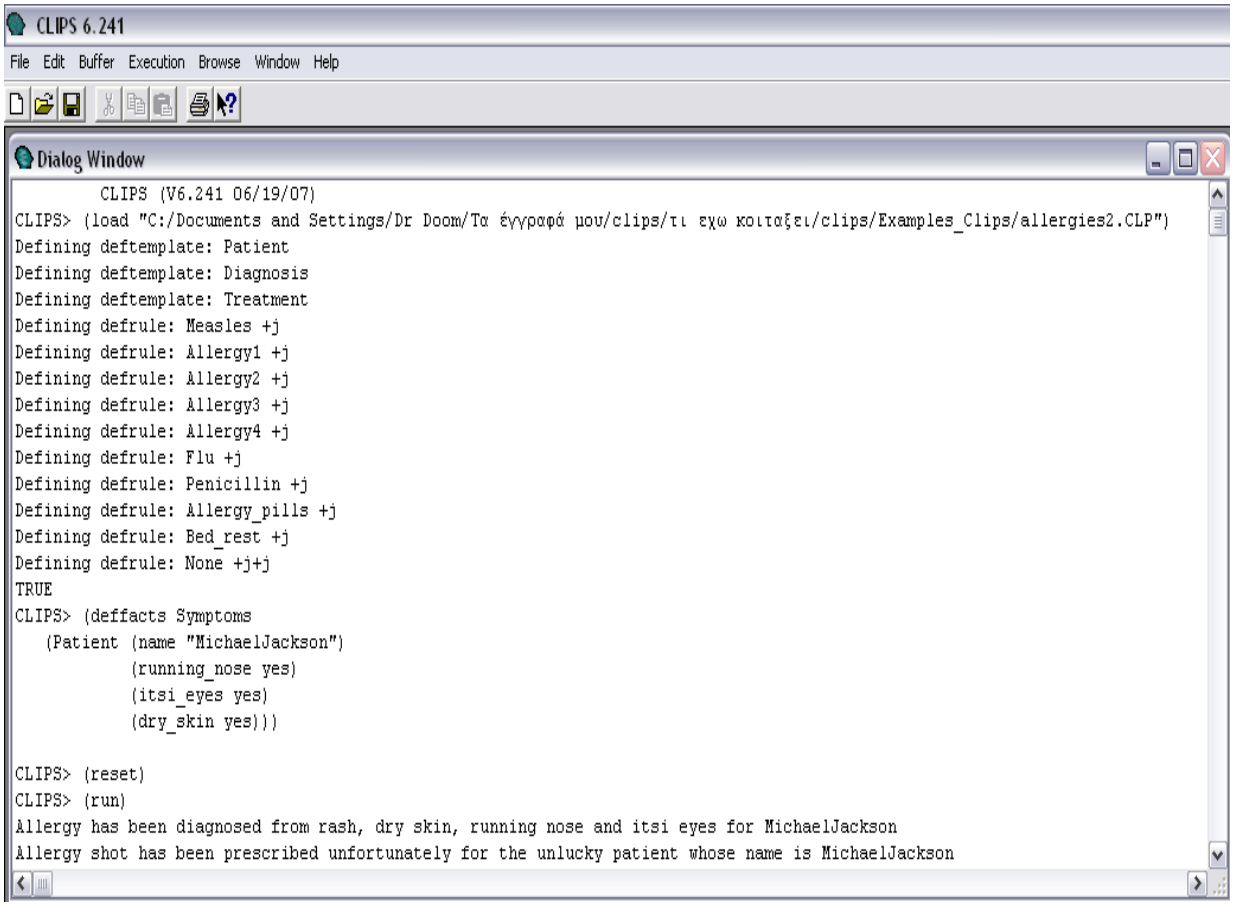
4.2.2 Δεύτερη περίπτωση: διάγνωση αλλεργίας

Έστω ένας ασθενής ο οποίος εμφανίζει αναφυλαξία, καταρροή μύτης, κνησμό ματιών και ξηροδερμία. Δημιουργείται και πάλι μια λίστα γεγονότων που θα περαστεί στην CLIPS με την εντολή (deffacts) με τα γεγονότα να αντιπροσωπεύουν τα συμπτώματα του ασθενή καθώς και το όνομα του:

```
(deffacts Symptoms
   (Patient (name "MichaelJackson")
            (running_nose yes)
            (itsi_eyes yes)
            (dry_skin yes)))
```

Κατά παρόμοιο τρόπο πληκτρολογείται (reset) για να εισαχθούν τα γεγονότα στην CLIPS και έπειτα (run) για να πάρουμε την πιθανή διάγνωση καθώς και την αντίστοιχη θεραπεία. Το έμπειρο σύστημα διέγνωσε ότι ο MichaelJackson πάσχει από αλλεργία εφόσον εμφανίζει τέτοια συμπτώματα

και προτείνει να του γίνει ένεση με αντιισταμινικά ή κορτικοστεροειδή φάρμακα προκείμενου να νιώσει καλύτερα (Εικόνα 4.2.1.2).



Εικόνα 4.2.1.2 Το έμπειρο σύστημα διέγνωσε ότι ο συγκεκριμένος ασθενής πάσχει από αλλεργία εξαιτίας της καταρροής μύτης, του κνησμού ματιών και της ξηροδερμίας και ότι η προτεινόμενη θεραπεία είναι η ένεση αντισταμινικών.

4.2.3 Τρίτη περίπτωση: δεν υπάρχει διάγνωση

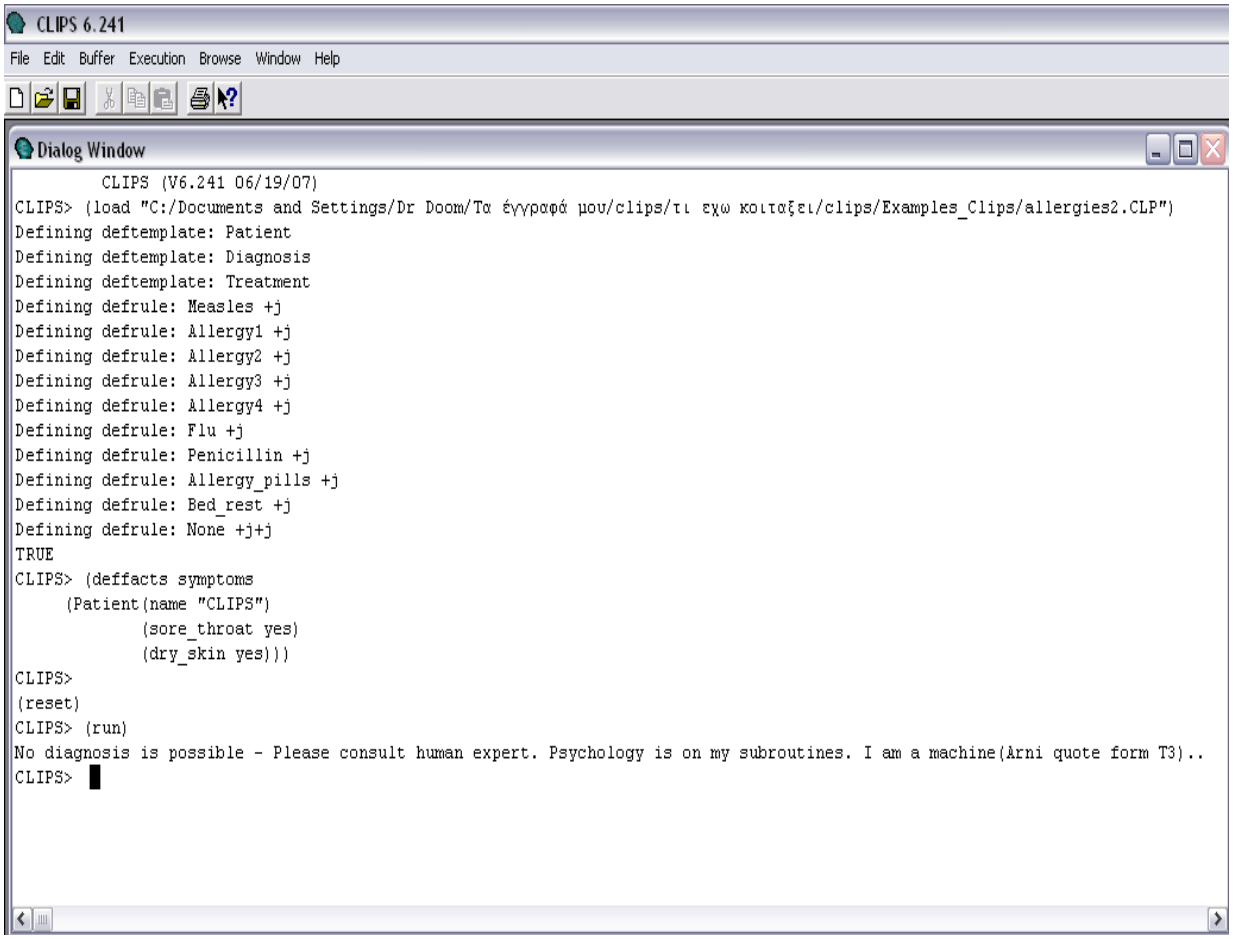
Έστω ένας ασθενής που λέγεται CLIPS. Ο CLIPS παρουσιάζει πονόλαιμο και ξηροδερμία. Δημιουργείται λοιπόν μια λίστα γεγονότων η οποία θα περαστεί στο έμπειρο σύστημα “D.A.S.2” προκειμένου να εξακριβωθεί τι αλλεργία έχει ο CLIPS.

```
(deffacts symptoms
     (Patient(name “CLIPS”)
             (sore_throat yes)
             (dry_skin yes)))
```

Το αποτέλεσμα της εισαγωγής αυτών των γεγονότων εφόσον έχει πληκτρολογηθεί (reset) και (run), είναι ότι δεν υπάρχει πιθανή διάγνωση για αυτόν τον συνδυασμό συμπτωμάτων με το ακόλουθο μήνυμα να εμφανίζεται:

"No diagnosis is possible – Please consult human expert. Psychology is on my subroutines. I am a machine(Arni quote from T3!..)"

και είναι προτιμότερο να αναζητήσει ο χρήστης την συμβουλή ενός ειδικού-συνήθως αλλεργιολόγου (Εικόνα 4.2.1.3).



Εικόνα 4.2.1.3 Το έμπειρο σύστημα δεν κατέληξε σε κάποια διάγνωση με βάση τα συγκεκριμένα συμπτώματα και προτείνει τον χρήστη να συμβουλευτεί έναν ειδικό εμπειρογνώμονα.

4.3 Συμπεράσματα- Συζήτηση

Μετά την παρουσίαση των αποτελεσμάτων παρατηρείται ότι ένα έμπειρο σύστημα διάγνωσης αλλεργιών και προτεινόμενων θεραπειών θα προσέφερε μεγάλη βοήθεια τόσο στους γιατρούς όσο και γενικότερα στο κλινικό προσωπικό. Το συγκεκριμένο σύστημα είναι ικανό να υποβοηθήσει στη λήψη απόφασης τους αλλεργιολόγους που θα έχουν μπροστά τους τη λίστα με τα πιθανά συμπτώματα της κάθε είδους αλλεργίας και να παρέχει τη δυνατότητα μέσω της εύκολης διαδικασίας των ερωτήσεων να βγάλει ο γιατρός διάγνωση ή να υποστηρίξει την προσωπική του εκτίμηση με ένα πολύ

εύκολο και προσβάσιμο τρόπο. Ωστόσο το σύστημα αυτό παρέχει και την δυνατότητα λήψης απόφασης στην περίπτωση ενός άπειρου ή αβέβαιου κλινικού προσωπικού εκπαιδεύοντας το παράλληλα με έναν πιο ομαλό και κατανοητό τρόπο από την τυπική εκμάθηση. Να σημειωθεί στο σημείο αυτό ότι ο ρόλος του έμπειρου συστήματος «D.A.S» είναι καθαρά συμβουλευτικός και σε καμία περίπτωση δεν αντικαθιστά την εμπειρία και τη σκέψη του γιατρού.

Το “D.A.S” υλοποιήθηκε με κύριο άξονα τον σχηματισμό κανόνων μέσω των οποίων θέτονται οι ερωτήσεις στον υποψήφιο χρήστη. Προκειμένου να γίνει αυτό χρησιμοποιήθηκαν μόνο πρότυπα γεγονότων και κανόνες όπως ακριβώς τους υποστηρίζει η CLIPS. Πληκτρολογώντας ένα απλό «ναι» ή «όχι» στις ερωτήσεις των συμπτωμάτων, δίνονται στοιχεία στο έμπειρο σύστημα ώστε να πραγματοποιήσει διάγνωση και να προτείνει αντίστοιχη θεραπεία. Με αυτή την υλοποίηση, το “D.A.S” είναι εύκολο στη μάθηση και στη χρήση καθώς αξιόπιστο και ευέλικτο χωρίς περιττές πολυπλοκότητες. Βεβαίως ο σκοπός δεν είναι να δημιουργηθεί ένα άψογο-σε επίπεδο κανόνων διάγνωσης - έμπειρο σύστημα καλύπτοντας κατά συνέπεια όλο το φάσμα των αλλεργιών στην ιατρική διότι κάτι τέτοιο θα ήταν εξαιρετικά χρονοβόρο και επίπονο. Στόχος ήταν να αποδειχτεί πως με τη κατάλληλη μελέτη και τον χειρισμό της γλώσσας CLIPS, μπορεί να σχεδιαστεί και να υλοποιηθεί ένα ολοκληρωμένο έμπειρο σύστημα διάγνωσης αλλεργιών το οποίο παράλληλα μπορεί να προτείνει συγκεκριμένες θεραπείες με βάση το είδος της αλλεργίας που έχει ο ασθενής μέσα από την πολύ απλή διαδικασία των ερωτήσεων.

Σε αυτό το σημείο να προστεθεί ότι όπως αναλύθηκε και στο υποκεφάλαιο 3.4 για την υλοποίηση του έμπειρου συστήματος “D.A.S” χρησιμοποιήθηκε και μια δεύτερη προσέγγιση μέσω της εισαγωγής γεγονότων στην CLIPS. Η διάφορα με τον πρώτο τρόπο υλοποίησης είναι η χρήση της εντολής (deffacts) με βάση την οποία μπορούν να εισαχθούν σε αυτή την έκδοση του συστήματος τα συμπτώματα που εμφανίζει ένας ασθενής ως ένας συνδυασμός γεγονότων. Μέσω των γεγονότων που εισέρχονται, το σύστημα βγάζει κατευθείαν την διάγνωση και τη θεραπεία χωρίς να είναι υποχρεωμένος ο χρήστης να απαντήσει σε όλες ή έστω σε κάποιες ερωτήσεις για να δει το αποτέλεσμα. Το μειονέκτημα σε αυτή τη περίπτωση είναι ότι ο υποψήφιος χρήστης είναι αναγκασμένος να πληκτρολογήσει τα δεδομένα του κάθε ασθενή προκείμενου να πάρει μια απάντηση. Για αυτόν τον λόγο υλοποιήθηκε το πρόβλημα μόνο με κανόνες ερωτήσεων ώστε να μην χρειαστεί να εισάγονται στο σύστημα όγκοι γεγονότων που χρειάζονται για να προκύψει μία διάγνωση. Ίσως να χανόταν κατά αυτόν τον τρόπο πολύτιμος χρόνος ενώ το να απαντά κανείς σε κάποιες ερωτήσεις είναι πολύ πιο εύχρηστο και πιο κατανοητό στον απλό χρήστη από το να πληκτρολογεί ο ίδιος τα γεγονότα. Βέβαια από την πλευρά του σχεδιασμού του προγράμματος μέσω CLIPS ορθώνεται μια δυσκολία η οποία ωστόσο περιορίζεται στη δημιουργία πολλαπλών κανόνων διάγνωσης. Όσο περισσότεροι κανόνες δημιουργηθούν που θα καλύπτουν όλους τους πιθανούς συνδυασμούς συμπτωμάτων αλλεργιών τόσο πιο αξιόπιστο και πιο πολύτιμο θα είναι το έμπειρο σύστημα.

Μετά την ολοκληρωμένη περιγραφή και παρουσίαση του έμπειρου συστήματος που ονομάστηκε “D.A.S” διαπυστώθηκαν κάποια γενικότερα συμπεράσματα σχετικά με τη χρησιμότητα των έμπειρων συστημάτων στην ιατρική τα οποία συνοψίζονται παρακάτω:

- Τα έμπειρα συστήματα είναι πολύ χρήσιμα στις περιπτώσεις όπου ένας ειδικός επιστήμονας απομακρύνεται από τον τομέα εργασίας του μαζί με τις πλούσιες γνώσεις που ενδεχομένως να κατέχει. Με τη χρήση των έμπειρων συστημάτων στην ιατρική μπορεί να αναπαρασταθεί αποτελεσματικά η γνώση των εμπειρογνωμόνων και να χρησιμοποιηθεί κατάλληλα. Τα έμπειρα συστήματα στον τομέα της ιατρικής έχουν ως στόχο είτε την υποστήριξη λήψης απόφασης του γιατρού ώστε να του υπενθυμίσουν τις διαθέσιμες επιλογές που υπάρχουν και τα θέματα που πρέπει να σκεφτεί είτε την λήψη απόφασης παρέχοντας έτσι βοήθεια σε προσωπικό χωρίς εμπειρία έχοντας παράλληλα και έναν εκπαιδευτικό χαρακτήρα. Στο συγκεκριμένο πρόβλημα μέσω του «D.A.S» διαχειρίστηκε η γνώση στο δύσκολο ζήτημα των αλλεργιών. Αυτό το σύστημα μπορεί μελλοντικά να αναβαθμιστεί εισάγοντας περισσότερη πληροφορία και να χρησιμοποιηθεί είτε για εκπαιδευτικούς σκοπούς είτε για υποβοήθηση στη λήψη απόφαση.

- Η CLIPS μπορεί να μην είναι τόσο διαδεδομένη ωστόσο είναι ένα καλό και εύκολο στην εκμάθηση εργαλείο σχεδιασμού έμπειρων συστημάτων. Μέσα από απλές εντολές και συνδυασμό αυτών, ο χρήστης μπορεί να δημιουργήσει μια εφαρμογή έμπειρου συστήματος πάνω σε πολλά ιατρικά θέματα που χρειάζονται υποστήριξη απόφασης ή λήψης απόφασης με αξιοπιστία και αποδοτικότητα.

- Ο συνδυασμός έμπειρων συστημάτων και γλώσσας CLIPS μπορεί να έχει εκπληκτική απόδοση εφόσον υπάρχουν γιατροί διατεθειμένοι να ασχοληθούν με τη χρήση τέτοιων συστημάτων στην κλινική πράξη καθώς και προγραμματιστές οι οποίοι να μπορέσουν να μοντελοποιήσουν τη γνώση των γιατρών στη μορφή ενός έμπειρου συστήματος. Εάν αναπτυχθούν και υλοποιηθούν κατάλληλα έμπειρα συστήματα στον τομέα της υγείας θα συμβάλλουν γενικότερα στη βελτίωση της παροχής των υπηρεσιών υγείας αφού το κλινικό προσωπικό θα έχει την δυνατότητα να εκπαιδεύεται μέσα από αυτές τις εφαρμογές ενώ οι γιατροί θα μπορούν να είναι σίγουροι για τις αποφάσεις τους ή ακόμα να αφιερώνουν τον πολύτιμο χρόνο τους σε δυσκολότερες περιπτώσεις ασθενών που απαιτούν εξειδικευμένη γνώση και εμπειρία, αφήνοντας τις απλές περιπτώσεις στα χέρια των έμπειρων συστημάτων. Ως συνέπεια ίσως στην ιατρική να μειωθούν σε σημαντικό ποσοστό τα ιατρικά λάθη και ο χρόνος εξυπηρέτησης των ασθενών βάζοντας τα θεμέλια για μια καλύτερη παροχή φροντίδας υγείας. Γιατί η υγεία είναι το πολυτιμότερο αγαθό του ανθρώπου και σκοπός των επαγγελματιών υγείας είναι η καλυτέρευση των ήδη υπαρχόντων συνθηκών ώστε κάποια στιγμή να μεγιστοποιήσουν τη παροχή και να βελτιστοποιήσουν τη ποιότητα των υπηρεσιών υγείας.

Βιβλιογραφία

1) K.S. Metaxiotis, J.-E. Samouilidis **(2000)** *Expert systems in medicine: academic illusion or real power?, Information Management & Computer Security, Volume 8, Issue 2, Page 75 – 79*

2) Robert Trowbridge, M.D.University of California, San Francisco School of Medicine, Scott Weingarten, M.D., M.P.H. University of California, Los Angeles School of Medicine **(2001)** *Making Health Care Safer: A Critical Analysis of Patient Safety Practices: Chapter 53, Clinical Decision Support Systems*

3) Joseph C. Giarratano, Ph.D. **(2002)** *Clips user's guide for the version 6.20.*

4) Peter Lucas **(2002)** *Expert Systems: A knowledge-based approach to intelligent systems*

5) E. Coiera **(2003)** *The Guide to Health Informatics (2nd Edition)* Arnold, London.

6) Χρυσόστομος Στύλιος **(2005)** *Έμπειρα συστήματα, διάλεξη στα πλαίσια του μαθήματος Έμπειρα Συστήματα στο ΤΕΙ Ηπείρου.*

7) Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Φ. Κόκκορας, Η. Σακελλαρίου **(2006)** *Παράρτημα 2. Το σύστημα κανόνων της Clips. Τεχνητή νοημοσύνη Β' έκδοση*.

8) Ιστοσελίδα της ηλεκτρονικής εγκυκλοπαίδειας Βικιπαίδεια, Πληροφορίες για την γλώσσα Clips
http://en.wikipedia.org/wiki/Clips

9) Ιστοσελίδα που περιέχει παραδείγματα στην γλώσσα Clips, Πληροφορίες για το σύστημα διάγνωσης αλλεργιών
http://www.cis.ysu.edu/~john/824/examples.html

10) Ιστοσελίδα που περιέχει αναλυτική περιγραφή αλλεργιών, Πληροφορίες για τα συμπτώματα των αλλεργιών
http://health.in.gr/allergies/default.asp

11) Ιστοσελίδα που περιγράφει γενικά τα έμπειρα συστήματα, Πληροφορίες για τα έμπειρα συστήματα
http://www.pcai.com/web/ai_info/expert_systems.html

12) Ιστοσελίδα που περιέχει άρθρα σχετιζόμενα με τα έμπειρα συστήματα, Πληροφορίες για τα έμπειρα συστήματα
http://www.openclinical.org/dssSuccessFactors.html

13) Ιστοσελίδα της ηλεκτρονικής εγκυκλοπαίδειας Βικιπαίδεια, Πληροφορίες για τα έμπειρα συστήματα
http://en.wikipedia.org/wiki/Expert_system

14) Ιστοσελίδα με περιγραφή των διαδικαστικών γλωσσών, Πληροφορίες για τις διαδικαστικές γλώσσες
http://www.techweb.com/encyclopedia/defineterm.jhtml?term=PROCEDURAL+LANGUAGE

15) Ιστοσελίδα με περιγραφή του Jess, Πληροφορίες για το Jess
http://herzberg.ca.sandia.gov/

16) Ιστοσελίδα της ηλεκτρονικής εγκυκλοπαίδειας Βικιπαίδεια, Πληροφορίες για το Jess
http://en.wikipedia.org/wiki/Jess_programming_language

17) Ιστοσελίδα της ηλεκτρονικής εγκυκλοπαίδειας Βικιπαίδεια, Πληροφορίες για την ασαφής Clips
http://en.wikipedia.org/wiki/FuzzyCLIPS